UNIVERSITY OF NOVI SAD

Development, implementation and theoretical analysis of the bee colony optimization meta-heuristic method

Dissertation written by

Tatjana Jakšić Krüger

Supervised by
Dr. Tatjana DAVIDOVIĆ

Faculty of Technical Sciences
2017

"Even dirty technology is good, if the intention is pure".
– Slavko Jakšić, 2012

In loving memory of my father.
You are still with me, real in memory as you were in flesh.

Mojoj majci Stanki Jakšić,
koja me je inspirisala.

Für meinen Ehemann Thomas Krüger,
den ich sehr liebe.

Sestri bliznakinji Tijani Jakšić Carević,
najboljoj sestri na svetu.

Acknowledgment

The first two years of my PhD studies were not connected with the topic of my current research. Back then, I was determined that my journey remain in the field of astronomy. However, my work with professor Tatjana Davidović on a problem of parallelization six years ago introduced me to this new chapter of my professional carrier. She proposed to me a collaboration that I both-handedly accepted and I have never doubted that it was the right choice. Learning about the field of operations research, a completely new one for me, was among the hardest things I had done until then. The dissertation reflects the stages of my development, from the basics of the field to its theoretical revelations. With the final revision I would like to thank everyone who has inspired me to learn and strive for discoveries which will contribute to the general knowledge of things.

First of all, I would like to thank my advisor professor Tatjana Davidović for her immense capacity to help and clarify both the smallest and the largest details about optimization problems. Your optimism, enthusiasm and the general concern for good science was what drew me to work with you. You are a good friend and colleague and an amazing scientist.

I would like to thank professors Joakim Lindblad, Milica Šelmić, Dragan Urošević Tibor Lukić and professor Jovanka Pantović for graciously accepting to be a part of the Committee. Thank you Joakim for your countless corrections of my style of writing. Thank you Milica for your thoroughness. Thank you Dragan for your insights and directions. Thank you Tibor for your calmness and understanding. Thank you Jovanka for your unyielding support.

In July 2016 I prepared the first draft of the thesis and I would like to thank professors Nataša Sladoje Matić and Joakim Lindblad who have helped me to improve the first version, to change the structure of my presentation and to envision a good thesis.

I would like to express my appreciation to the projects' leaders at the Mathematical Institute of the Serbian Academy of Sciences and Arts for their support and encouragement.

Finally, I would like to express my deepest gratitude to my husband, my family and friends for their love and support. Without you, the thesis would have not been possible. I want to thank my friends Anatolij Mihajlov, Attila Cséki and Petar Maksimović for their objectivity and the help to stay focused.

Abstract

Difficult optimization problems are problems that require computationally intensive resources to generate optimal or sub-optimal solutions. Therefore, development of efficient optimization methods is essential. The contribution of stochastic methods, such as meta-heuristics, to dealing with difficult optimization problems is widely recognized in the operations research community. Current topics in the field of optimization address the success of stochastic methods through empirical or theoretical study, or both.

The subject of this dissertation is the study of the Bee Colony Optimization (BCO) meta-heuristic method. The study follows three lines of research: theoretical analysis of convergence, implementation of parallelization strategies and empirical analysis of performance. We deliberate our research goals from the point of the design of the BCO algorithm.

A growing interest in the convergence properties of the BCO method has inspired the main segment of the thesis. Theoretical analysis of BCO implies mathematical verification of the asymptotic convergence towards an optimal solution, assuming certain conditions. We identify these conditions for constructive (BCOc) and improvement BCO (BCOi). Based on the recently published tutorial we inspect two types of convergence, the *best-so-far* and the sophisticated *model* convergence. The best-so-far convergence has just to answer if the optimal solution is reachable by the considered search method, i.e., will the desired optimum be found if the algorithm is given enough time. The model convergence analyzes the properties of the algorithm that direct the search process toward the subspace containing the optimal solution. Regarding the best-so-far convergence, we prove that BCO already satisfies necessary and sufficient conditions. To fulfil the conditions of the model convergence the BCO algorithm must incorporate the global knowledge and the specific modification/construction rules. We prove that BCO is well founded and provides a good basis for model convergence.

Operating with population of solutions makes BCO suitable for parallelization. Therefore, we explore potential parallelization strategies for distributed memory multiprocessor architecture, as it allows utilization of a large number of processors. Three parallelization strategies and five corresponding coarse-grained implementations are considered under the Message Passing Interface (MPI) communication protocol. The first strategy involves independent execution of various BCOc algorithms. The second strategy is related to synchronous cooperative execution of the BCOc algorithms. The third strategy defines a novel general approach that implements asynchronous cooperation between the algorithms. The strategy enables diversified search suitable for various multiprocessor topologies. We propose two variants of asynchronous strategy: with centralized and with non-centralized communication control. The results demonstrate that all parallel BCO algorithms provide excellent performance compared to the sequential implementation. The independent executions enable significant speedup while preserving the quality of solutions. As for both of cooperative strategies, the

quality is improved in addition to speeding up the computations. Moreover, the asynchronous strategy outperforms the synchronous with respect to both solution's quality and running time. We observe that the best performance is achieved by engaging a modest number of processors (up to 12). Several reasons might cause this behavior: communication overhead between a larger number of processors and/or stochastic nature of the BCO method that corrupts a systematic behavior of the parallel implementations.

We address the empirical study of BCO(c|i) by conducting experimental tests in domains where sophisticated alternatives already exist in order to perform sensitivity analysis of algorithm's response to changes in problem and parameter settings. Our objective is to contribute to the further development of the BCO algorithms. Statistical and visual analysis are two general directions that we exploit, supported by an off-line strategy. The design of our studies is founded on averaging response values and comparing profile data between the factors of interest. The two studies are focused on the influence of two actual and two novel BCO parameters on the measured outcomes. They encompass comparisons between several heuristic algorithms and the selection of the best candidate for construction/modification rules embedded in the BCO(c|i) algorithm. Statistical analysis helps to establish the statistical and practical significance between the best reported results of investigated algorithm instances. Visual analysis is employed to provoke research questions about the performance and to extend the information obtained by statistical analysis. Specifically, the focus is on the influence and interactions between a subset of BCO parameters: population size (B), control number of constructions/transformations (NC), choice of objective function (ME) and loyalty function (L_f). Both statistical and visual analysis are conducted for BCOc while we employ only visual analysis for the BCOi solver.

Contents

Acknowledgment i

Abstract iii

Module I Preliminaries

Chapter 1 Introduction **3**
1.1 Current topics in the field of meta-heuristics 3
1.2 Motivation and goals of the thesis 4
1.3 Structure of the thesis 6
1.4 Publications and contributions 8
1.5 Chapter summary 10

Chapter 2 Optimization problems and methods **11**
2.1 Introduction to optimization 11
2.1.1 NP-hard problems 12
2.1.2 Optimization problems 15
2.2 Optimization methods 20
2.2.1 Background 20
2.2.2 Classification 21
2.2.3 Heuristic methods 23
2.3 Examples of optimization problems and their methodologies . . 24
2.3.1 Scheduling problems 25
2.3.2 Methods for $P||C_{max}$ 29
2.3.3 Satisfiability problem 30
2.3.4 k-SAT solvers 32
2.4 Chapter summary 35

Chapter 3 Meta-heuristic methods **37**
3.1 Introduction to meta-heuristics 37
3.1.1 Classification of meta-heuristics 38
3.1.2 Nature- and bio-inspired methods 40
3.1.3 Swarm intelligence 41
3.2 Examples of meta-heuristics 42
3.2.1 Simulated annealing 43
3.2.2 Evolutionary computation and genetic algorithm 44
3.2.3 Ant Colony Optimization 45
3.2.4 Particle swarm optimization 46
3.2.5 Artificial Bee Colony 47

3.2.6 Tabu search . . . 47
3.2.7 Variable neighborhood search . . . 49
3.2.8 Final remarks . . . 50
3.3 Chapter summary . . . 50

Chapter 4 Bee colony optimization method . . . 53
4.1 The development of BCO . . . 53
4.1.1 Bees in nature . . . 53
4.1.2 BCO model . . . 55
4.1.3 The evolution of BCO . . . 56
4.2 The BCO Algorithm . . . 57
4.2.1 Pseudo-code for BCO . . . 59
4.2.2 Variants of the BCO algorithm . . . 60
4.2.3 Backward pass and loyalty functions . . . 63
4.2.4 Comparing ABC, ACO and BCO . . . 73
4.2.5 Final remarks . . . 74
4.3 Chapter summary . . . 75

Module II Methodology and contributions

Chapter 5 Theoretical analysis of the asymptotic convergence of the BCO method . . . 79
5.1 Motivation for theoretical analysis . . . 79
5.2 Instance- and model-based algorithms . . . 81
5.3 Theoretical background . . . 82
5.3.1 Best-so-far convergence . . . 84
5.3.2 Model convergence . . . 86
5.4 Convergence analysis of the BCO method . . . 87
5.4.1 Approximation of an optimal solution . . . 88
5.4.2 Generic BCO algorithm . . . 89
5.4.3 Various cases of the BCO algorithms . . . 89
5.5 Best-so-far convergence of BCO . . . 90
5.6 Model convergence of BCO . . . 92
5.6.1 Preliminary conditions for model convergence . . . 93
5.6.2 Model convergence of BCOc . . . 94
5.6.3 Model convergence of BCOi . . . 100
5.7 Final remarks . . . 101
5.8 Chapter summary . . . 102

Chapter 6 Parallelization strategies for the BCO algorithm . . . 103
6.1 Motivation for parallelization of meta-heuristics . . . 103
6.2 Parallelization strategies of meta-heuristics . . . 104
6.2.1 Parallelization of ABC . . . 105
6.2.2 Parallelization of ACO . . . 106
6.3 Parallelization strategies of the BCO algorithm . . . 108
6.3.1 Independent execution of the BCO algorithms . . . 108
6.3.2 Synchronous cooperation of the BCO algorithms . . . 110

6.3.3 Asynchronous cooperation of the BCO algorithms 111
6.4 Comparison of the results for parallel BCOc executions 113
6.4.1 Experimental environment 113
6.4.2 Test instances 115
6.4.3 Comparison of independent BCOc executions 115
6.4.4 Comparison of cooperative executions 117
6.5 Final remarks 121
6.6 Chapter summary 122

Chapter 7 Methodology of experimental study of BCO 125
7.1 Motivation for empirical analysis 125
7.2 Experimental study of meta-heuristics 126
7.2.1 What is an experiment? 126
7.2.2 Measure of performance 126
7.2.3 Configuration methods 128
7.2.4 Types of parameters 129
7.2.5 Sensitivity analysis 129
7.2.6 Reproducibility 130
7.2.7 Statistical methods 130
7.3 Experimental analysis of BCO 133
7.3.1 Motivation of the BCO study 133
7.3.2 Structure of the BCO study 134
7.3.3 Categorizations of BCO parameters 135
7.3.4 Hierarchy diagram of BCO parameters 135
7.3.5 Experimental setup for BCO analysis 137
7.3.6 Final remarks 138
7.4 Chapter summary 139

Chapter 8 Development and empirical analysis of BCOc 141
8.1 Sensitivity analysis of the BCOc algorithm 141
8.1.1 Research goals 142
8.1.2 Test instances 143
8.1.3 The first BCOc for $P||C_{max}$ 144
8.2 Candidate heuristics for $P||C_{max}$ 144
8.2.1 Experimental evaluation of candidate heuristics 145
8.2.2 Conclusions regarding the best heuristic 147
8.3 Development of the BCOc algorithm for $P||C_{max}$ 148
8.3.1 Experimental methodology 149
8.3.2 Design of BCOc forward pass 150
8.3.3 Design of BCOc backward pass: qualitative parameters . 151
8.3.4 Collection of results 152
8.4 Screening BCOc parameters: basic plots 159
8.4.1 3-D plots: qualitative parameters 159
8.4.2 2-D plots: quantitative parameters 162
8.5 Statistical analysis: hypothesis testing 166
8.6 Stopping criterion: N_{it} 167
8.6.1 Case study: the best loyalty function 168
8.6.2 Case study: the effect of ME 172

8.7 Stopping criterion: CPU time . . . 175
8.7.1 Case study: the best loyalty function . . . 175
8.7.2 Case study: the effect of ME . . . 177
8.8 Recruitment dynamics . . . 179
8.9 Final remarks . . . 183
8.10 Chapter summary . . . 184

Chapter 9 Development and experimental analysis of BCOi . . . 187
9.1 Sensitivity analysis of the BCOi algorithm . . . 187
9.1.1 Research goals . . . 188
9.1.2 Swarm intelligence for SAT . . . 188
9.1.3 Problem instances . . . 189
9.1.4 Experimental methodology . . . 191
9.2 Candidate heuristics for 3-SAT . . . 193
9.2.1 Experimental evaluation of candidate solvers . . . 194
9.2.2 Final remarks and conclusions for candidate heuristics . . . 196
9.3 Development of BCOi for 3-SAT . . . 197
9.3.1 Design of the BCOi algorithm . . . 197
9.3.2 Backward pass of `randBCOi` and `WalkBCOi` . . . 198
9.4 Empirical study of `randBCOi` . . . 199
9.4.1 Case study: `randBCOi` for uf100-430 . . . 199
9.4.2 Case study: `randBCOi` for uf50-218 . . . 203
9.4.3 Conclusions for `randBCOi` . . . 204
9.5 Experimental study of WalkBCOi . . . 204
9.5.1 Case study: Evaluation function ev_1 . . . 204
9.5.2 Case study: Evaluation function ev_2 . . . 207
9.5.3 Conclusions for `walkBCOi` . . . 208
9.6 Final remarks . . . 208
9.7 Chapter summary . . . 208

Chapter 10 Conclusions and future work . . . 213
10.1 Concluding remarks . . . 213
10.2 Future work . . . 215

Bibliography . . . 217

Appendix A Complementary material . . . 247
A.1 Empirical study of BCOc . . . 247
A.1.1 Time measurement . . . 247
A.1.2 Box-plots . . . 247
A.1.3 Candidate heuristics for $P||C_{max}$ . . . 247
A.1.4 Comparative study of candidate heuristics . . . 251
A.1.5 Experimental evaluation of the best heuristic: `sLPT+BF` . . . 253
A.2 The BCO algorithm . . . 257
A.2.1 BCOc for $P||C_{max}$: calculate constructive moves . . . 257
A.2.2 Choice of reference case: heuristic vs. $NC = 1$ . . . 258
A.3 Statistical analysis of BCOc . . . 259
A.3.1 Preliminaries . . . 259

A.3.2 Repeated measures ANOVA 261
A.3.3 Ranking R procedure for ME 264
A.4 Empirical study of BCOi 265
A.4.1 SATLIB 265
A.4.2 Number of transformations 266

Appendix B Tables and graphics 267
B.1 Empirical study of BCOc 267
B.1.1 Analysis of the bestfit heuristic 267
B.1.2 BCOc: response landscape for two methods of evaluation 267
B.1.3 Comparison of 3-D surface plots 267
B.1.4 Case study: Maximal allowed CPU time 271
B.2 Tables and Figures 272
B.2.1 Performance table 272
B.3 Bar-chart of instances 306

List of Figures

2.1 Partial solution for TSP . . . 24
2.2 Gantt diagram–schedule of tasks on machines (taken from [Dav12] . . . 29
2.3 Main procedure of Schöning's algorithm. . . . 33
2.4 GWSAT . . . 34
2.5 WalkSAT from [Sel94]. . . . 35

3.1 Pseudo-code of the SA algorithm . . . 43
3.2 Pseudo-code of the EA algorithm. . . . 44
3.3 Pseudo-code of the ACO algorithm. . . . 45
3.4 Pseudo-code of the PSO algorithm. . . . 46
3.5 Pseudo-code of the ABC algorithm. . . . 47
3.6 Pseudo-code of the TS algorithm. . . . 48
3.7 Pseudo-code of the VNS algorithm. . . . 50

4.1 Waggle dance (courtesy of Wikimedia, CC-BY-SA-3.0) [Jü11]. . . . 54
4.2 Recruiters and uncommitted bees. . . . 58
4.3 Illustration of the process of recruitment. . . . 59
4.4 Pseudo-code for BCO . . . 60
4.5 An example of partial solutions after the first forward pass (taken from [Dav12]) . . . 61
4.6 An example of partial solutions in the second forward pass after first move (taken from [Dav12]). . . . 61
4.7 An illustration of the s-th forward pass (courtesy of Tatjana Davidović), [Dav11a] . . . 62
4.8 Possible result of a recruiting process within s-th backward pass, (courtesy of Tatjana Davidović), [Dav11a] . . . 62
4.9 An example of complete solutions after $(s+1)$-th forward pass . . . 63
4.10 Influence of forward pass counter and normalized quality of solutions on the loyalty function $p_b^{0,u}$ . . . 66
4.11 Influence of O_b on the loyalty function p_b^1 . . . 67
4.12 Influence of u and O_b on the loyalty function $p^{4,u}$ . . . 68
4.13 Influence of u and O_b on the loyalty function $p^{5,u}$ . . . 69
4.14 Influence of u and O_b on the loyalty function $p^{6,u}$ . . . 70
4.15 Influence of u and O_b on the loyalty function $p^{7,u}$ . . . 70
4.16 Influence of forward pass counter on the loyalty function p^8 . . . 71
4.17 Influence of u and O_b on the loyalty function $p^{9,u}$ . . . 71
4.18 Probability based choice (courtesy of Tatjana Davidović and Milica Šelmić). 72
4.19 Roulette wheel (courtesy of Tatjana Davidović). . . . 72

6.1 Independent execution of several BCO algorithms. . . . 109

6.2 Classification of parallelization strategies for BCO. The colored fields indicate settings of our implementations. 112
6.3 Network Topology. 114
6.4 Improvement in time of the current best solution, during CBCO with a fixed number of communications 118
6.5 Improvement in time of the current best solution, during CBCO with a variable number of communications 119

7.1 Hierarchy of parameters in the BCO algorithm design. 136

8.1 Box-plot for $m = 12, 16$ and 9 instances in relation to the n, where possible outliers are marked with red crosses. 143
8.2 Response landscape of 10 BCOc with $\min(ev_1)$ for Iogra100_12, $N_{it} = 100$.160
8.3 Effects of parameters ME, Lp and the problem instance characteristics. Profile lines with squares denote problem instances in the class $m = 12$ and with triangles the class $m = 16$. Different colors refer to value of n. Stopping criterion is $N_{it} = 100$. 161
8.4 Representative graphic of influence of three BCO parameters (*ME*, *LF*, *NC*) on solution quality for $N_{it} = 100$. Graphic represents changing of average solutions over values of NC for each loyalty function, on problem instance Iogra100_12. For easiness of comparing, relative error is used. 163
8.5 Influence of three methods of evaluation and 10 loyalty function on the performance of BCOc for $P||C_{max}$, when $NC \in [1, 100]$ and fixed B that generated the best results. Stopping criterion is maximum number of iterations, $N_{it} = 100$. 164
8.6 Influence of three methods of evaluation and 10 loyalty function on the performance of BCOc for $P||C_{max}$, $NC \in [1, 100]$ and fixed B that has reported the best solution. Stopping criterion is allowed CPU time $T = n/100[s]$. 165
8.7 Graphic of propagation of best results generated by each method of evaluation when maximal number of iterations is provided. Dotted line corresponds to performance of `sLPT+BF`. 175
8.8 Graphic of propagation of best results generated by each method of evaluation when maximal CPU time is provided. Dotted line corresponds to performance of `sLPT+BF`. 179
8.9 Propagation of variable R_e (average number of recruiters per experiment) over $NC \in [1, 100]$ for $B = 10$ and instance *Iogra100_12*. 180
8.10 Propagation of variable R_{miss} (average number of misses per experiment) reported for $B = 10$, $NC \in [1, 100]$ and test instance *Iogra100_12*. 182
8.11 Propagation of variable R_{max} (average number of times condition $R = B$ has occurred in backward pass per experiment) reported for $B = 10$, $NC \in [1, 100]$ and test instance *Iogra100_12*. 182
8.12 Grouping of loyalty functions regarding the similarities exhibited during the recruitment process. 183

9.1 Semi-log plots of cumulative distribution of $\overline{n_{flip}}$ for three 3-SAT solvers and three problem sets. The x-axis shows $\overline{n_{flip}}$ based on 1000 runs/instance and stopping criterion $T = 5s$. 195
9.2 Pseudo-code for BCOi for satisfiability problem. 198
9.3 Algorithmic structure of the evaluation function ev_2 within BCOi. 199
9.4 Matrix-plots of average number of unsatisfied clauses generated by BCO with the Class I functions: $p^{1,2,8}$ and $p^{3,n_{iter}}$. 200
9.5 Matrix-plots of average number of unsatisfied clauses generated by different BCOi instances and Class II loyalty functions for *uf100* problem-set. 202
9.6 Evolution of average number of flips for different `randBCOi` instances. Problem instances belong to class uf50-218. Stopping criteria is MAXFLIPS$=10^6$ 206
9.7 Evolution of average number of flips for different `walkBCOi` instances and the evaluation function $ev_1 = numfalse$. Problem instances belong to class uf100–430. Stopping criteria is MAXFLIPS$=10^6$. 209
9.8 Evolution of average number of flips for different `walkBCOi` instances and the evaluation function ev_2 . Problem instances belong to class uf100–430. Stopping criteria is MAXFLIPS$=10^6$. 210

A.1 Pseudo-code for `sLPT+BF` 249
A.2 Pseudo-code for `sLPT+FF` 250
A.3 Pseudo-code for `sLPT+ES` 251
A.4 Pseudo-code for `sLPT+sES` algorithm. 252
A.5 Propagation of solutions for different instances generated by `sLPT+BF` after 100, 200, 300, 400, 500 and 1000 iterations. Solid line corresponds to class of instances when number of machines is 12, and dashed line to 16 machines. 257
A.6 Graphic of propagation of the average quality of solution generated by standalone heuristic and of BCO algorithm when $NC = 1, B = 20$. Stopping criteria is N_{it}. 259
A.7 Pairwise comparison between results of the best performing loyalty function (here p_b^2), and the rest of loyalty functions within the same group, on test instance Iogra100_16. 263

B.1 Occurence of best found solution of each run generated with `sLPT+BF` for different instances, when $N_{it} = 1000$, $m = 12$. 268
B.2 Occurrence of best found solutions of each run generated with *sLPT+bestfit* for different instances, when $N_{it} = 1000$, $m = 16$. 268
B.3 Occurence of best found solution of each run generated with `sLPT+BF` for different instances, when $N_{it} = 10000$, $m = 12$. 269
B.4 Occurrence of best found solutions of each run generated with *sLPT+bestfit* for different instances, when $N_{it} = 10000$, $m = 16$. 269
B.5 Response landscape of 10 BCOc with $\max(ev_1)$ for Iogra100_12, $N_{it} = 100$. 270
B.6 Response landscape of 10 BCOc with $\max(ev_2)$ for Iogra100_12, $N_{it} = 100$. 271
B.7 Landscape of measured outcomes for four different loyalty functions, when $N_{it} = 100$, $min(ev_1)$ and problem instance *Iogra100_12*. 272
B.8 Bar chart for Iogra100 306
B.9 Bar chart for Iogra150 307
B.10 Bar chart for Iogra200 308

B.11 Bar chart for Iogra250 309
B.12 Bar chart for Iogra300 310
B.13 Bar chart for Iogra350 311
B.14 Bar chart for Iogra400 312
B.15 Bar chart for Iogra450 313
B.16 Bar chart for Iogra500 314

List of Tables

6.1 MBCO Scheduling results – test problems Iogra100_12 and u250_4. . . . 116
6.2 CBCO Scheduling results – test problem Iogra100_12. 117
6.3 CBCO Scheduling results – test problem u250_04. 120
6.4 GBCO Scheduling results – test problem Iogra100_12. 121
6.5 GBCO Scheduling results – test problem u250_04. 122
6.6 Comparison between sequential and parallel BCOc within the same CPU time – test problem Iogra100_12. 123

7.1 Statistical tests. 132

8.1 Scheduling results for test problems with known optimal solutions appropriated from [Dav06b] with $n = \{100, 250\}$, $n_{run} = 100$, $N_{it} = 100$ and different number of machines. 146
8.2 Minimal number of iterations needed to obtain value 811 for test instance Iogra100_12 and $n_{run} = 100$. Time is reported in seconds. 147
8.3 Parameter space for experimental analysis of BCOc. 153
8.4 Response values after a run of the BCOc instance. 153
8.5 Overview of the response values and the corresponding descriptive statistics of the experiment. 154
8.6 Best and worst average solutions found by corresponding BCOc algorithms for problem instance *Iogra100_12* [Dav06b]. Stopping criterion, $N_{it} = 100$. 155
8.7 Best and worst average solutions found by corresponding BCOc algorithms for problem instance *Iogra100_16* [Dav06b]. Stopping criterion, $N_{it} = 100$. 156
8.8 Best and worst average solutions found by corresponding BCOc algorithms for problem instance *Iogra100_12* [Dav06b]. Stopping criterion, $T = 0.1[s]$. 157
8.9 Best and worst average solutions found by corresponding BCOc algorithms for problem instance *Iogra100_16* [Dav06b]. Stopping criterion, $T = 0.1[s]$. 158
8.10 Repeated-measure ANOVA and Friedman's test results for equivalence of means between loyalty functions of specific method of evaluation for $\alpha = .05$ and maximum number of iterations. 169
8.11 Results of RMANOVA and Friedman's test for N_{it}. 170
8.12 Results of pairwise comparison test ($N_{it} = 100$): list of loyalty functions that are not significantly different from the control group. 171
8.13 Table of best results and its corresponding parameter configurations for class of $m = 12$ of problem instances. Stopping criterion, $N_{it} = 100$. . . 173

8.14 Table of best results and its corresponding parameter configurations for class of $m = 16$ of problem instances. Stopping criterion, $N_{it} = 100$. . . 173
8.15 Results of RMANOVA and Friedman's test for T. 176
8.16 Results of pairwise comparison test ($T = n/100[s]$): list of loyalty functions that are not significantly different from the control group. 176
8.17 Table of best results and its corresponding parameter configurations for class of $m = 12$ of problem instances. Stopping criterion: CPUtime. . . 178
8.18 Table of best results for $m = 16$. 178

9.1 SATLIB 3-SAT satisfiable instaces. 191
9.2 Results of `wsat` for uf100-430 [Hoo98a]. 193
9.3 Descriptive statistics for n_{flip} reported by three candidate solvers for three problem instances in class uf50-218, $T = 5[s]$. 195
9.4 Results for `Rand`, `SchRand` and `WalkSAT` for SATLIB problem-sets. Optimality criterion is an average number of flips. 196
9.5 Parameter space for experimental analysis of BCOi on 3-SAT. 198
9.6 Results of comparison between `Rand` and the best configurations of `randBCOi` for SATLIB problem-set uf100-430. Optimality criterion is N_{uns}. 201
9.7 Results for `randBCOi` for problem-set uf50-218. Optimality criterion is N_{flip}. 203
9.8 Descriptive statistics for `walkBCOi` with ev_1 for problem-set uf100-430. Optimality criterion is N_{flip}. 205
9.9 Descriptive statistics for `walkBCOi` with ev_2 for problem-set uf100-430. Optimality criterion is N_{flip}. 207

A.1 Scheduling results - test problems with known optimal solutions adopted from [Dav06b] with $m = 12, 16$, $n_{run} = 100$ and a different number of iterations ($N_{it} = 100, \ldots, 1000$). 254
A.2 Scheduling results - test problems with known optimal solutions adopted from [Dav06b] with $m = 12, 16$, $n_{run} = 100$ and different number of iterations ($N_{it} = 2000, \ldots, 10000$) (cont). 255

B.1 Repeated-measure ANOVA and Friedman's test results for equivalence of means between loyalty functions of specific method of evaluation for $\alpha = .05$ and maximum CPU time. 273
B.2 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra150_12* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 274
B.3 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra150_16* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 275
B.4 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra200_12* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 276
B.5 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra200_16* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 277
B.6 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra250_12* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 278
B.7 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra250_16* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 279

B.8 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra300_12* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 280
B.9 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra300_16* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 281
B.10 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra350_12* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 282
B.11 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra350_16* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 283
B.12 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra400_12* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 284
B.13 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra400_16* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 285
B.14 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra450_12* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 286
B.15 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra450_16* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 287
B.16 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra500_12* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 288
B.17 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra500_16* [Dav06b]. Stopping criterion, $N_{it} = 100$. . . . 289
B.18 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra150_12* [Dav06b]. Stopping criterion, $T = 0.15[s]$. . . 290
B.19 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra150_16* [Dav06b]. Stopping criterion, $T = 0.15[s]$. . . 291
B.20 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra200_12* [Dav06b]. Stopping criterion, $T = 0.2[s]$. . . 292
B.21 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra200_16* [Dav06b]. Stopping criterion, $T = 0.2[s]$. . . 293
B.22 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra250_12* [Dav06b]. Stopping criterion, $T = 0.25[s]$. . . 294
B.23 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra250_16* [Dav06b]. Stopping criterion, $T = 0.25[s]$. . . 295
B.24 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra300_12* [Dav06b]. Stopping criterion, $T = 0.3[s]$. . . 296
B.25 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra300_16* [Dav06b]. Stopping criterion, $T = 0.3[s]$. . . 297
B.26 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra350_12* [Dav06b]. Stopping criterion, $T = 0.35[s]$. . . 298
B.27 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra350_16* [Dav06b]. Stopping criterion, $T = 0.35[s]$. . . 299
B.28 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra400_12* [Dav06b]. Stopping criterion, $T = 0.4[s]$. . . 300
B.29 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra400_16* [Dav06b]. Stopping criterion, $T = 0.4[s]$. . . 301
B.30 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra450_12* [Dav06b]. Stopping criterion, $T = 0.45[s]$. . . 302

B.31 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra450_16* [Dav06b]. Stopping criterion, $T = 0.45[s]$. . . 303
B.32 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra500_12* [Dav06b]. Stopping criterion, $T = 0.5[s]$. . . 304
B.33 Best and worst average solutions found by corresponding BCOs for problem instance *Iogra500_16* [Dav06b]. Stopping criterion, $T = 0.5[s]$. . . 305

Module I

Preliminaries

CHAPTER 1

Introduction

This chapter provides motivation and structure of the dissertation. It begins with an overview of actual topics in the field of meta-heuristic methods that has inspired our research. In Section 1.2 we elaborate on these topics and establish general research questions of the dissertation. Finally, the structure of the thesis is given along with associated publications.

A meta-heuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms to efficiently produce solutions of high quality [Voß12, Sör13]. In practice, it is customary to use the term *solution* for any result that an algorithm reports, regardless of its quality. Therefore, a *high quality solution* is a result that we consider to be *close enough* to the true solution of a problem, usually called *optimum*. Moreover, in optimization a *suboptimal* solution is commonly regarded as a local optimum. Population-based meta-heuristics are methods that incorporate an idea of combining existing solutions in order to generate new one(s). The idea has proven successful and has inspired development of various meta-heuristic methods. The subject of this thesis is the method inspired by principles of bee's colony behavior and their collaboration mechanism during foraging process in nature, namely Bee Colony Optimization method (BCO). Our work follows two directions of research: theoretical and empirical. Theoretical analysis is exploring convergence properties of BCO towards global optimal solution using probability analysis. The empirical part consists of two segments. The first is focused on improving performance of the BCO algortihm by implementing parallelization strategies. The second deals with empirical evaluation of BCO's performance by analyzing the influence of its parameters on the measured outcomes.

1.1 Current topics in the field of meta-heuristics

Theoretical and empirical analysis of optimization methods, as well as the development of parallelization strategies, represent current topics in the field of operations research (OR). The development of natural computation (such as evolutionary computing, swarm intelligence, neural networks) has established a new paradigm as to how problems may be solved, significantly increasing the number of new meta-heuristic methods. During the last three decades we can follow an expansion of the field of population-based meta-heuristic methods, as they provide opportunity to compare more solutions at the same time. Various empirical and theoretical studies have demonstrated the advantage of meta-heuristics as methods that exhibit flexibility and robust-

ness to structure of optimization problems, while providing high quality solutions at a reasonable computational time. Given that meta-heuristics are characterized by their parameters, proper configuration brings improvement in their overall performance.

BCO is a meta-heuristic method that has proved to be very efficient, being applied in a wide portfolio of optimization problems. It was proposed by Panta Lučić and Dušan Teodorović at the beginning of the 21^{st} century, and since then has been used by researchers from all over the world. Lučić and Teodorović stand among the first who used the basic principles of collective bee intelligence to solve optimization problems [Luč01, Luč02b, Luč03a]. BCO is still being extensively developed and explored with numerous successful applications [Dav11a, Dav12, Eda08, Mar07, Nik13a, Šel10, Teo05, Teo08, Teo06, Teo07, Won10b]. A recent survey of BCO can be found in [Dav15b, Teo15]. The inspiration for creating the multi-agent system, such as BCO, came from the analogy between the natural behaviour of bees while looking for food and behaviour of optimization algorithm searching for an optimal solution of an optimization problem.

There are several contributions that we want to present in the dissertation. The first central topic of this thesis concerns theoretical analysis of BCO, as we provide sufficient and necessary conditions under which asymptotic convergence of BCO toward the optimal solution holds. The practical segment of this thesis examines properties of BCO which provide a good basis for parallelization and propose several strategies for parallelization of BCO on various levels. We aim to contribute to this topic since, to the best of the author's knowledge, only a few basic ideas regarding parallel execution of BCO were reported in the literature. Empirical analysis of BCO is addressing sensitivity analysis of BCO to changes in its parameter configurations and the structure of problem instances. We introduce a term *loyalty function* as a characterization of the probability decisions inside BCO, and demonstrate the influence of several new and existing loyalty functions on the values of decision probabilities. New evaluation functions are proposed for two hard combinatorial problems: simple variant of the scheduling problem and 3-SAT. The practical contributions we provide have opened a new chapter in the study of BCO. Namely, we have increased the total number of the BCO parameters that may be configured before the algorithm's execution. We address the issue of configuration by following two general directions: statistical and/or visual analysis.

1.2 Motivation and goals of the thesis

Many real-life problems are difficult with respect to the number of their appropriately definable factors. A substantial part of these problems can be represented as optimization problems. The main objective of an optimization problem is to find an optimal solution, preferably, in some timely manner. The difficult optimization problem is the problem that cannot be solved to optimality or no method can guarantee a bound in reasonable time [Flo09, pg. 2309]. Classical approach in the fields of operations research and mathematical programming is to employ methods, such as *exact methods*, that guarantee to find an optimal solution. Their primary feature is a systematic search of the solution-space or an intelligent way that reduces the solution-space by some preference. However, the exact methods can express inefficiency with respect to both time and memory for various difficult optimization problems. The issue has motivated the

researches to address the search efficiency in other ways. One way, and also the central topic of the dissertation, is to invoke stochastic processes in the search for high quality solutions. These methods, known as *non-deterministic*, are often more efficient than the exact methods for problems that are: multimodal, non-differentiable, NP-hard, etc. [Whi70, Smu12]. We are particularly interested in *meta-heuristic* methods, as a class of general computational methods that commonly applies some non-deterministic rules. Numerous meta-heuristic methods have been developed up to this time as they provide solutions of high quality for a reasonable execution time.

Intensive applications of meta-heuristic methods have initiated the research on their empirical and theoretical analysis. Despite the usefulness and practical efficiency of meta-heuristics, there exists a serious drawback related to the actual quality of generated solution(s). Whether meta-heuristics can find an optimal solution cannot be proved without the knowledge or without the utilization of an exact method. In particular, if the optimal solution is not reachable, it is hard to determine how far the generated solution might be from it. We have addressed this issue by conducting theoretical analysis of the necessary steps required for the successful implementation of BCO that would assure asymptotic convergence of the BCO algorithm to the desired optimal solution. We provide sufficient and necessary convergence condition and prove that, when satisfied, BCO can solve an optimization problem if given enough time.

Apart from the significant advances in computer technology and progress in disciplines relevant for solving optimization problems, some problems are still challenging in the sense that it is hard to solve realistically large problem instances within a reasonable computation time. A possible approach for dealing with the issue is parallelization. One of the goals of parallelization is to produce a linear speedup of algorithm's execution on distributed- or shared-memory processor machines. However, parallelization of non-deterministic algorithms is usually not as straightforward as it is for deterministic algorithms due to their stochastic component, therefore, are more challenging. In the dissertation we propose parallelization strategies of BCO for distributed memory multiprocessor systems and examine the benefits of its parallel execution.

The performance of meta-heuristics are closely connected to their particular applications and implementations, which is why their analysis is mainly conducted experimentally. Meta-heuristics are characterized by their parameters, producing another optimization problem when searching for the most appropriate parameter configurations. Advancement in automatic algorithm configuration (e.g., off-line parameter tuning) has provided a helpful framework for evaluation of meta-heuristic algorithms. However, these strategies might be computationally expensive to oversee the performance of an algorithm on the complete (or large part of) parameter space. Furthermore, it is highly recommended to become familiar with the underlying procedures of the tuning methods, especially when dealing with some specific requirements. Accordingly, the time spent to find an appropriate tuner and to adjust its environment can be spend to design a study that will incorporate only the necessary parts. Without loss of generality, we conduct an empirical analysis of the BCO algorithm by means of a comparative study between a large number of BCO's parameter configurations. In particular, we study the sensitivity of BCO to changes in its parameter values by incorporating various suggestions from the literature, while pertaining an overview on the large part of the parameter space. The significance of our work is to provide reliable values of the BCO parameters since the sub-optimal parameter values can lead to stag-

nation and an extremely high variability during the execution [Hoo07, pg. 12]. We are primarily interested in the performance of BCO while changing the maximal number of bees and the number of constructive/improvement moves within an iteration. However, due to the increasing number of proposals for loyalty function, the research topic has expanded towards modular BCO parameters, such as loyalty and evaluation function. Modular suggests that they are not necessary functions, but can be a subroutine (module). Experimental analysis of BCO is conducted for two combinatorial problems (scheduling and satisfiability) by employing *off-line* (*static*) *configuration* of the BCO parameters. For the scheduling problem, we revisit the choice of an underlying heuristic to provide some empirical evidence about the most influential part of BCO that ensures convergence towards an optimal solution. Additionally, for both considered problems we propose new evaluation functions and study which, among the loyalty functions, exhibits the best behavior w.r.t. the quality of response values and their influence on the convergence speed. The motivation is to step away from the conventional BCO parameter settings and provide rationale behind the selection of its values. In addition, we investigate some of the properties of benchmark problem instances of scheduling problems [Dav06b] and revisit the critical values for the 3-SAT instances. Finally, we discuss the underlying mechanisms of BCOc exploring the recruitment process, however, for fixed configurations of the BCO parameters.

1.3 Structure of the thesis

The thesis is divided into ten chapters, organized within two modules. The first module provides introduction to the field of optimization with the focus on meta-heuristic methods and BCO. The second module outlines three topics of the dissertation: theoretical analysis, parallelization strategies and empirical study. The topics are arranged within separate chapters. The corresponding chapters in the second module begin with a formal background and related work. The contributions of our study are presented in the second half of the corresponding chapters.

Chapter 1(current chapter): We elaborate the main motivation and research objectives and introduce the structure of the dissertation.

Chapter 2: Optimization problems and methods. The chapter provides an introduction into the optimization by reviewing the general definition of optimization problems. Definition of the non-linear optimization problem helps establish necessary notation used in this thesis. A short survey of different optimization problems and methods is given next. A special class of optimization methods (heuristic methods) is covered, as they represent an integral part of the most meta-heuristics. As an illustration, we provide description of two combinatorial problems: scheduling of independent tasks to identical machines and the special variant of satisfiability problem, 3-SAT. For each problem, a short survey of widely used heuristic methods is given.

Chapter 3: Meta-heuristic methods. The chapter is devoted to the topic of meta-heuristic methods. First, we provide definitions of class of mathematically- and nature-inspired meta-heuristic methods, with special attention to the swarm intelligence methods. Secondly, for each class we describe several most common meta-heuristics.

Chapter 4: Bee colony optimization method. The chapter is dedicated to BCO, its development and implementations. It reviews a behavioral model of bees in nature

used as inspiration for development of the BCO method. Two variants of the BCO algorithm are revisited and new concept introduced – the *loyalty function*. We discuss impact of 10 different loyalty functions on the performance of the BCO algorithm.

Chapter 5: Theoretical analysis of the asymptotic convergence of the BCO method. The chapter is dedicated to the theory of BCO. It begins with the theoretical background of convergence analysis of meta-heuristics and a survey of the existing literature on this topic. The chapter covers formal definitions of probability analysis, used as the basic tool in study of meta-heuristics. We focus on the parts that are integral for our theoretical conclusions. The second part of the chapter presents results of the theoretical analysis of the BCO method. Following the guidelines presented in [Gut09] we investigate two types of convergence for both variants of BCO: BCOc and BCOi. Sufficient and necessary conditions that establish the best-so-far and the model convergence are presented for six different scenarios w.r.t. methods of selection. A part of the chapter originates from [JK14a, JK14b, JK16b].

Chapter 6: Parallelization strategies for the BCO algorithm. The chapter is dedicated to the study of parallelization strategies for BCO. In the first half we review the literature that deals with parallelization of meta-heuristics for distributed memory architectures. It provides examples of successful parallel implementations on two nature-inspired population-based meta-heuristic methods. In the second half of the chapter we present five parallel implementations of the BCO algorithm. In the experimental segment of the chapter we present comparison results for these implementations. The conclusion is that invoking modest number of processors (up to 12) generates best results. Majority of the chapter originates from [Dav13].

Chapter 5: Methodology of experimental study of BCO. In this chapter we briefly review the literature on empirical analysis of meta-heuristic methods and discuss several examples of measure of performance. We establish experimental conditions used for empirical analysis of the BCO method. A brief inspection of common statistical tools in the operations research is also given.

Chapter 8: Development and empirical analysis of BCOc. An exhaustive empirical study of the BCOc algorithm for the problem of scheduling independent tasks on identical machines is presented. In order to choose best heuristic for the BCOc algorithm, in the first segment of the chapter we conduct several experimental comparisons and graphical evaluations. Next, we address the issues of *tuning* of both qualitative and quantitative BCO parameters. The main objective is to test BCOc in domains where sophisticated alternatives already exist in order to investigate its robustness and performance. A part of the chapter originates from [JK16c].

Chapter 9: Development and experimental analysis of BCOi. The chapter provides results of an exhaustive empirical study of the BCOi algorithm developed for 3-SAT. The structure of the chapter is similar to the previous one. However, we consider only visual representation of our results. A part of the chapter originates from [JK16a].

Chapter 10: Conclusions and future work. The thesis concludes with final remarks and conclusions of presented material. Moreover, some ideas for future research are provided.

Appendix A: Complementary material provides an additional information regarding the two variants of the BCO algorithm. However, the majority of material is focused on results of BCOc. We present the procedure for determining the number of constructive moves (adding components to partial solution). We give results of comparison

study between the *reference case* of the BCOc algorithm (BCO with only one bee) and the underlying heuristic. We demonstrate that the two exhibit almost identical behavior, and thus, performance of underlying heuristics can be replaced by results of the reference case.

Appendix B: Tables and graphics provides figures and tables of the results of BCOc, related to Chapter 8.

1.4 Publications and contributions

During research activities prior to the dissertation, two lines of research were followed. The first relates to increasing the effectiveness and efficiency of BCO variants for instances of a particular optimization problem, by studying the influence of BCO parameters on the overall performance and implementing different parallelization strategies. The second line of the research covers the theoretical analysis of the BCO convergence properties. The main questions of the dissertation have been elaborated within several journal and conference papers, which are covered here. Results are specified according to the order in which they were published.

- Tatjana Davidović, **Tatjana Jakšić**, Dušan Ramljak, Milica Šelmić and Dušan Teodorović. **Parallelization strategies for bee colony optimization based on message passing communication protocol**. Optimization, Vol. 62, Iss. 8, 2013.

Development of the new and the efficient parallelization strategies for BCO represents the first practical contribution of the dissertation. BCO operates on a population of solutions, therefore, presumes good basis for parallelization. Few basic ideas regarding the parallel execution of the BCO algorithm have been reported in [Dav11b]. Due to its stochastic nature, parallelization of the BCO algorithm for distributed memory architectures was considered to be challenging. Therefore, we propose three parallelization strategies for a distributed memory multiprocessor architecture under the Message Passing Interface (MPI) communication protocol. The first strategy involves independent execution of several BCO algorithms, and was used in the early work on parallelization of BCO [Dav11b]. The above mentioned paper contains a synchronous strategy implementing cooperation between various BCO algorithms and defines more general approach. Another novelty is the development of asynchronous strategies introduced to provide a more diversified search. Asynchronous strategies are implemented in two ways: with centralized and with non-centralized communication control. The presented experimental results, addressing the problem of static scheduling of independent tasks on identical machines, show that the parallel BCO algorithms provide superior performance compared against the sequential implementation. Specifically, in the case of independent executions, a significant speedup was obtained while preserving the quality of solution. As for both of cooperative strategies, the solution quality was improved. However, both asynchronous strategies outperform the synchronous with respect to both solution quality and the running time. Another interesting conclusion is that in order to obtain better quality solutions within the same amount of wall–clock time, only a modest number of processors should be engaged in the parallel BCO execution. There are a couple of reasons why this is the case. Firstly, the overhead

from communication between multiple processors prevails over benefits of parallelization. Secondly, the stochastic component of the BCO method might influence the loss of a systematic speedup, thus progressing as a random walk. In addition, as the time and the quality of solutions are relatively stable, the conclusion is that adding new processors does not improve an overall performance. This is covered in greater detail in Chapter 6.

- **Tatjana Jakšić Kruger**. O konvergenciji metaheurističke metode optimizacije kolonijom pčela. Zbornik IV Simpozijuma "Matematika i primene", pp. 176–188, 2014.
- **Tatjana Jakšić Krüger**, Tatjana Davidović. **Model convergence properties of the constructive Bee Colony Optimization algorithm**, in Proceedings of the 41th Symposium on Operations Research (SYM-OP-IS 2014), pp. 340-346, Divčibare, Serbia, Sep. 16-19, 2014.
- **Tatjana Jakšić**, Tatjana Davidović, Dušan Teodorović, Milica Šelmić. **The Bee Colony Optimization Algorithm and its Convergence**. International Journal of Bio-Inspired Computation, Volume 8, Issue 5, pp. 340–354, 2016 (accepted 2014).

The lack of theoretical understanding of BCO inspired publications of three research papers, listed above. The aim was to understand the mechanisms of probability decisions incorporated in the BCO method and analyze its influence on the convergence toward optimal solution. Based on the tutorial [Gut09] we consider two types of convergence, *best-so-far* convergence and the *model* convergence. We provide necessary and sufficient conditions under which BCO is converging in the best-so-far sense. In order to assure the model convergence of BCOc we present sufficient properties. Specifically, we conclude that BCOc converges, under certain restrictions, with probability one toward the optimal solution. These restrictions imply that BCOc needs to implement learning mechanism used for adaptation of probability rule for selecting a (partial) candidate solution. We prove that BCOc is well founded and provides a good basis for model convergence. Results are elaborated in Chapter 5.4 and in the above mentioned papers.

- **Tatjana Jakšić Krüger**, Tatjana Davidović. **Empirical Analysis of the Bee Colony Optimization Method on 3-SAT problem**, Proceedings of the 43th Symposium on Operations Research (SYM-OP-IS 2016), pp. 297–301 Tara, Serbia, Sept. 20-23, 2016.
- **Tatjana Jakšić Krüger**, Tatjana Davidović. **Sensitivity analysis of the Bee Colony Optimization Algorithm**, in Proceedings of the 7th International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2016), pp. 64-80, Bled, Slovenia, May 18-20, 2016.

Recent contribution to the thesis relates to the empirical analysis of the constructive and the improvement version of BCO on two problems: problem of scheduling independent tasks to identical machines and 3-SAT, respectively. The empirical analysis has been conducted to investigate the sensitivity and robustness of BCO to changes in the parameter configurations and problem instances. The results have been obtained by a detailed inspection of different parameter configurations, such as number of bees,

number of forward/backward passes, the choice of evaluation and loyalty functions. The case studies have been conducted for the fixed interval of the parameter B and for the fixed number of constructive/modification moves during an execution of the BCO algorithm. Results of the experiments are presented in chapter 8 and 9 while the part has been published in the mentioned conference paper.

Contributions that have not been published and are for the first time elaborated in the dissertation may be found in Chapters 4, 5, 7, 8, 9. Material in Chapters 2 and 3 should be considered as a contribution to the curriculum in the field of optimization. Content of these chapters is inspired by the substantial amount of valuable tutorials, lectures and books, however, it is our understanding that their main focus is often on the specific topics. We have tried to consider various aspects of the field of optimization by reviewing its history, recalling the existing definitions and reorganize some of the current classifications.

1.5 Chapter summary

In this chapter we present motivation and objectives of the thesis. More precisely, we:

- Introduce research topics of this dissertation.
- Provide general definitions of the *high quality solution* and meta-heuristic methods.
- Emphasize the significance of working with population of solutions at the same time.
- Review our publications and we provide the short summaries of the contributions.

CHAPTER 2

Optimization problems and methods

The chapter contains a general introduction to the field of optimization. It provides definitions of hard optimization problems and the most common optimization methods. To establish necessary notation, utilized throughout the dissertation, we review the definition of (nonlinear) optimization problem. Next, we briefly survey different types of optimization problems. Section 2.2 contains a historical synopsis on the development of optimization methods and their classification, followed by a short survey of various heuristic methods. Finally, we review two combinatorial optimization problems (the problem of scheduling independent tasks to identical machines and 3-SAT) and the corresponding heuristic methods important for our study of BCO.

2.1 Introduction to optimization

Optimization simply means to find the best solution or to operate a system in the most effective way [Hau04, Wan09]. It can be considered as a process of adjusting the input in order to attain the optimal (minimal or maximal) system output. Optimization today has an essential place in the real-life and science, as optimization problems can arise from anywhere. It became essential for the design and the analysis of complex systems (e.g., management of the freight and passenger transportation and traffic) which has introduced problems such as: infrastructure planning, timetable generation, route planning, multi-modal transport optimization, etc. Other real-life problems might be nurse scheduling problem, multiple sequence alignment problems or prediction of protein structure and recognition, string selection and comparison problems, vehicle routing problems, etc. The complexity of the problems grows daily with respect to the input size and the constraints imposed on the data. Consequently, optimal solutions to these problems are increasingly hard to obtain.

During the last decades the successful applications and implementations of various optimization methods may be observed. A motivation behind their development is to maximize the efficiency in terms of speed and/or memory usage while faced with the growing needs of the modern industry. Especially if the problem exhibits a huge number of local extremes, such as a Griewank's function, uneven distribution of extremes, deception extremes, high-dimensional problems, etc. [Spa03][Fod12, Chapter 4]. With the development of algorithms, such as branch-and-bound (B&B) and integer liner programing (ILP) during the 70s, the border for number of solvable optimization problems has expended. Nevertheless, many optimization problems required innovations, why the meta-heuristics have received the attention from the research community. We de-

tect again a new trend of research in the field of optimization, with the rise of so called *matheuristics*. They represent hybrids between meta-heuristic and exact methods.

Most of the real-life optimization problems belong to the class of *NP-hard problems* [Pap98]. For this reason, the following section is dedicated to the formal definition of hard problems from the aspect of the theory of computational complexity.

2.1.1 NP-hard problems

Typical problems encountered during bachelor studies in computations, belong to the class of problems that are solvable in polynomial time. Even the simple real-world optimization problems like determining the shortest path from work to home, while taking detours to visit a couple of places before reaching the final destination, are intuitively considered as easy. Our basic comprehension can, therefore, deceive us, especially when faced with current scientific and industry problems. To describe a hard optimization problem, we recapitulate its definition from the the theory of computational complexity. Because the definition relies on the concept of decision problems, the aim of this section is to point out relation between search and decision variant of the considered optimization problem.

Decision problems are problems for which the answer for every valid input is YES or NO. Examples of such problems are: satisfiability problem (SAT, given a Boolean formula, is it satisfiable), graph coloring (can a graph of n vertices be colored using k colors, such that no two adjacent vertices are colored with the same color, where $k < n$), halting problem (given an algorithm and its input, will it ever complete the computations), traveling salesperson problem (also known as traveling salesman problem, TSP, given a number of cities n, an $n \times n$ matrix of nonnegative integers $[d_{i,j}]$ that represent distances between cities, is there a tour τ of length less than or equal to given integer l, where tour τ is a cyclic permutation of cities), and many others [Pap98]. *Search problems* are problems which require identification of solutions from a countable or infinite set of possible solutions. The majority of above mentioned decision problems can be used to formulate the corresponding search problems, where in this case we want to have more information than simple yes or no. For graph coloring, a search problem would correspond to finding pairs of $(vertex, color)$ which satisfy the conditions of the problem, i.e., that two vertices are not colored with the same color (in case these pairs exist). In similar fashion, the search problem of TSP is easy to formulate as the objective is to find the actual tour of a length less then or equal to l. We can also identify *verification problems* in which we need to demonstrate if provided solution really is a solution of the given problem. To demonstrate, previous examples are used. In the case of graph coloring, a verification problem would be that for a given pair $(vertex, color)$ we need to verify if it is a solution, or not. In the case of TSP, for some given tour, we want to verify if it is a cyclic permutation with the length not larger than l.

Regarding the optimization problems, we distinguish between their search and decision variants. The search variant of an optimization problem corresponds to finding a solution that optimizes a given objective function. For example, finding the shortest tour of a TSP, or a valid coloring with minimal number of colors in the graph coloring problem. Decision variant of an optimization problem implies providing an answer to a question: does the optimal (feasible) solution of the corresponding optimization

problem exists.

A significant portion of research in computer science is dedicated to construction of algorithms that solve problems in an efficient manner. We say that an algorithm is useful in practice, *efficient*, *polynomial bounded*, or *tractable* if its running time is guaranteed to be bounded by a polynomial $p(n)$ in the size n of the input of the problem. In other words, an algorithm is known to be of *polynomial-time* complexity if, given any instance as input, it generates a correct answer in at most $c \cdot n^k$ steps, where k and c are constants and n is the size of the instance, or the number of bits needed to specify it. Polynomial boundedness has been shown to have both theoretical and practical significance [Law76]. A formal definition of these types of the problems are in the theory of computation known as class P. The class P is defined as the class of all problems for which there exists a deterministic polynomial-time algorithm. Even if parameter k, from definition of tractable problems, takes values such as $k = 50000$, is enough to meet the requirement of this definition. The examples of the problems in class P are: sorting, minimum cut, minimum spanning tree and shortest paths [Gar79, Gol88].

Exponential time algorithms are algorithms whose time complexity function cannot be bounded by a polynomial $p(n)$. They are usually considered *inefficient*, although a few proved to be very useful in practice (e.g. simplex algorithm for linear programming, branch-and-bound for knapsack problem). The inefficiency is revealed when considering a solution of the large problem instances, reflected by the time complexity functions provided in [Gar79, pg. 6]. For example, observing time complexity function n^5, where n denotes an input length, an increment in the size of the instance by only 10 increases the computing time by a factor 4, whereas for 2^n the computing time is increased 2^{10} times. A problem is considered to be *intractable* if there is no guaranty that it can be solved in polynomial time, and therefore may take exponential time to discover a solution [Gar79]. In the worst case, there are problems for which no algorithm can solve them, often refered to as being *non-computable* or *undecidable* (non-computable problem that requires a yes/no answer). One such example is the *halting* problem, described by Turing in [Tur36], [Gar79, pg. 12]. One can also consider intractability with respect to the space (memory requirements). It occurs when the solution cannot be described with an expression having length bounded by a polynomial function of the input size. A good example of such problem is to generate all possible permutations of a set of n integers. The length of the output, in this case, cannot be a polynomial function of the length of the input, thus it cannot be computed in polynomial time [Kam09].

The question then is, which problems have polynomial time algorithms? Unfortunately, this is not an easy question. Since almost all real problems are intractable, the main resort is to find as efficient algorithms as possible in terms of the running time and the memory usage. There are many tools one can use to approach such problem. To estimate complexity of the algorithm a hypothetical computer is considered: it has unlimited random access memory, the access time to memory is constant and unaffected by the size of the data and the number of data stored, it is capable of executing instructions such as integer arithmetic operations, numerical comparisons, branching operations, that always require one unit of time [Law76]. An example of such a machine is widely used Turing machine [Gar79].

Class NP is defined as a more general than class P, i.e., it is a class of all problems for which some previously provided solution can be verified in polynomial time [Gar79].

NP refers to the *nondeterministic polynomial* problem, that is, problems that are solvable in polynomial time on a nondeterministic Turing machine (Cook-Levin theorem [Coo71]). Various formal definition exists in the literature related to theory of computational complexity [Gar79, Gol99, Cor01, Ogn04, Pap07]. There is an alternative theory known as *probabilistically checkable proof*, PCP, also used to provide analysis of computing *approximate* solutions to NP problems [Aro98]. It implements a computational model as probabilistic Turing machine in order to measure and formalize the performance of probabilistic algorithms.

To define what means hard problem for NP class of problems, let us first recall the standard definition: a problem $\mathcal{P}$ is *hard* for a class of problems C if every problem in C can be reduced to $\mathcal{P}$. Therefore, no problem in C is harder than $\mathcal{P}$, since an algorithm for $\mathcal{P}$ can solve any problem in C. The notion of hard problems depends on the type of reduction being used. If problem $\mathcal{P}$ also belongs to the class C, then $\mathcal{P}$ is said to be *complete* for C [Cor01].

This all leads us to the important class of NP-complete problems[(1)]. The foundations for the theory of NP-completeness were presented by Stephen Cook [Coo71]. He displayed the importance of *polynomial time reducibility*, which states that a polynomial time reduction from one problem to another ensures that any polynomial time algorithm for the second problem can be converted into a corresponding polynomial time algorithm for the first problem. Another significant result Cook provided is that one particular problem in NP, called the *satisfiability problem* (SAT), has the property that every other problem in NP can be polynomially reduced to it. This means that if we can solve SAT problem efficiently, then we can solve all other problems in NP efficiently as well. Equivalently, if any problem in NP is intractable, then the SAT problem also is intractable. Therefore, the satisfiability problem is the hardest problem in NP [Gar79]. Richard Karp proved that there is a great number of other NP problems that share this property of being as hard as SAT. This equivalence class which consists of the hardest problems in NP is called *NP-complete class* of problems. There are hundreds of known NP-complete problems [Gar79].

Here, we demonstrate the connection between decision and search problems. It can be shown that for all NP-complete problems if there exists an efficient procedure for solving a search problem, there also exist an efficient procedure for solving the corresponding decision problem [Gol99, Lecture 1, pg. 4]. We concentrate in the rest of the thesis on the search problems. Search problems are considered to be a general class of problems and a decision problem may simply be considered as a special type of the corresponding search problem [Gar79]. However, to classify optimization problems regarding the theory of computational complexity, a step into generalized world of problems is required.

Search problem is NP-hard if it is as least as hard as some NP-complete problem. Class of NP-hard problems, therefore, introduces broader set of problems, the one for which verification of correctness in polynomial time doesn't have to exist. For additional formal definitions a reader is referred to [Gar79, Ogn04, Pap07]. The accent here is on the conclusion from the literature about NP-hard problems: any NP-hard problem, to which we can transform an NP-complete problem, cannot be solved in

(1) The presented material on NP-complete problems follows the guidelines of the lectures, given by prof. dr Tim Roughgarden, *Algorithms: Design and Analysis, part 2*, (http://theory.stanford.edu/ tim/videos.html).

polynomial time unless $P = NP$ [Gar79].

As already mentioned, a search variant of an optimization problem can be formulated as a series of decision problems. To elaborate further on this correspondence, let us observe an instance of the traveling salesperson problem that is specified by some n vertices (cities), distances amongst all of the pairs of vertices as positive integers and a bound $l \in \mathbb{Z}^+$ [Gar79, pg. 18]. The corresponding decision problem would be: is there a tour that has a total length no more than l? Considering this decision problem, possibly exponentially many times, it is possible to find the optimal solution for related search variant of TSP. Furthermore, if the objective function of the search problem is relatively easy to evaluate, then the *decision problem can be no harder than the the corresponding optimization problem*. Therefore, if the decision variant of TSP is NP-complete, than the search variant is at least as hard [Gar79, pg. 19]. The same reasoning can be applied to essentially all search problems. Therefore, we may state that an optimization problem is NP-hard if the corresponding decision problem is NP-complete [Gar79, Pap91b][Bru07, pg. 46].

2.1.2 Optimization problems

Optimization maintains an essential place in practical and scientific world. Today, common optimization problems are versatile, e.g.,: sequence alignment problems, a prediction of protein structure, genome rearrangement problems, string selection and comparison problems, vehicle routing problems, air traffic scheduling problem, etc. Therefore, we begin with a brief survey of the development of the optimization problems, followed by their formal definitions taken from optimization theory. Finally, we provide the classification of optimization problems.

2.1.2.1 History and definitions

The word *optimum* comes from a Latin word *optimus* meaning to describe the property of being the best, ideal or perfect. The essence of the word has not changed, and today designates a condition that produces the best result. What we consider the best in some practical problem can be described in mathematics as an extreme of a suitable mathematical function. Therefore, the process of searching for configurations, or states of physical processes, that are the best among a given set of finite or infinite alternatives is called *optimization*.

First historical endeavor in mathematical formalization of optimization problem can be tracked back to the beginnings of the field of mathematical analysis called *calculus of variations*, when in the 17th century a Brachistochrone problem was first proposed. The problem was described by John Bernoulli as a requirement to find a path along which a particle (point-like body) would move under the influence of gravity, from one point to another in the least amount of time. But it was only until middle of 1940s when the optimization methods, such as linear and dynamic programming, were first used [Dut04]. Despite the fact that optimization problems have been always a major part of our every-day lives, from the invention of wheel (to minimize the drag during transportation) to maximizing the monetary profits, it was only in the last 60 years when the optimization has grown as an independent field.

The fundamental elements of a mathematical model of optimization problems are: (*i*) a set of objects (variables), each with an associated contribution, (*ii*) an objective

function used for computing the value of a particular subset or order of objects, and (*iii*) a set of constraints, i.e., the feasibility rules specifying how subsets/orderings may be built [Cra14]. The goal of an optimization method is to assign possible values to the variables so as to acquire the optimal value of the objective function, while satisfying all constraints. General definition of optimization problems is as follows [Dav06a]:

Definition 1. Let $f : S \to \mathbb{R}$ be real function, defined on the set S, and let $\mathcal{X} \subseteq S$ be some given set. Find

$$\min f(\boldsymbol{x}),$$

under condition (constraint)

$$\boldsymbol{x} \in \mathcal{X}. \qquad \diamond$$

To further demonstrate what each of this element means, we review the definition of the *(nonlinear) programming problem* from [Pap98]. The formulation of the problem is suitable for the introduction of basic terminology.

Definition 2. (Nonlinear) programming problem consists of finding a(all) value(s) for the decision vector $\boldsymbol{x} \in \mathbb{R}^n$, as to

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) \geq 0,\ i = 1, \ldots, m \\ & h_j(\boldsymbol{x}) = 0\ j = 1, \ldots, p. \end{array} \tag{2.1}$$

$\diamond$

The components x^i of vector $\boldsymbol{x} = (x^1, x^2, \ldots, x^n)^T$ are called *decision variables*, while f, g_i and h_j represent general functions of these variables. Function $f(\boldsymbol{x})$ signifies the *objective function*, and $g_i(\boldsymbol{x})$ and $h_j(\boldsymbol{x})$ denote *constraint functions* of the optimization problem. When considering merely objective function $f(\boldsymbol{x})$, the decision vectors $\boldsymbol{x}$ are called *possible solutions*, while a set of all possible solutions is called a *search space* [(2)], denoted here as S. Therefore, function $f(\boldsymbol{x})$ is a mapping $f : S \to \mathbb{R}$, where $S \subseteq \mathbb{R}^n$. Furthermore, the set of possible solutions $\boldsymbol{x}$ that meet all the constraints of functions $g_i(\boldsymbol{x})$ and $h_j(\boldsymbol{x})$ is called a *feasible region* ($\mathcal{X}$) and its elements *feasible solutions*. Based on the provided terminology, the (nonlinear) programming problem can now be defined as finding the minimum of function $f(\boldsymbol{x})$ on the set $\mathcal{X}$. In other words, we are searching for some $\boldsymbol{x}^*$ that satisfies:

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}),\ \ \forall \boldsymbol{x} \in \mathcal{X}. \tag{2.2}$$

Feasible solution $\boldsymbol{x}^* \in \mathcal{X}$ that satisfies condition (2.2) is called (*global*) *optimal solution*. The best (optimal) solution is the one satisfying feasibility rules with the property that the corresponding value of the objective function is minimal among all possible combinations. A feasible solution which is not optimal is called *sub-optimal* [Pap98, Dav06a]. The solutions that are optimal within a subset of a feasible region are called *local optima*. Formal definition of local optimum is given in Definition 4 in the text to follow. It is worth noting that maximization of $f(\boldsymbol{x})$ can be described as the problem of minimization of $-f(\boldsymbol{x})$, therefore, it is enough to consider one type of optimization problems.

[(2)] Yang in his book [Yan10] has distinguished the notion of search space from so-called *solution space* stating that the search space is a complete space $\mathbb{R}^n$, and that solutions space represent the domain of objective function f.

The essential part of the optimization problems is the objective. The objective should be considered as an expression in form of an unambiguous (mathematical) statement of the goals to be achieved [Mic04]. In operations research, the statements typically concern minimization or maximization of an objective function, usually under some restrictions. Optimization expressions, derived from Definition 2, are standardized in the optimization community. When it comes down to implementation of an optimization method, a slightly different terminology may be used [Hoo05]. First of all there is a notion of a *candidate solution* defined as a potential solution that may be encountered during the attempt to solve given problem instance, but may not satisfy all the conditions from the problem definition. Moreover, candidate solutions satisfying the logical conditions can be called feasible or *valid* [Hoo05, pg. 16]. Furthermore, the search space is defined as a set of all candidate solutions of a problem instance, and the *solution set* as the set of all valid solutions. Other expressions, such as *neighborhood relation*, *memory states*, *initialisation function* and *step function* are also used [Hoo05] [Gon07, chapter 14]. In the recent literature, the expression *candidate solution* is employed with a slightly different meaning than in [Hoo05]. It is assumed that candidate solutions are always feasible, and sometimes they belong to the subspace of the so-called promising solutions that should be visited first during the search. Henceforth, we refer to the later definition in the thesis.

Another useful distinction between theory and practice is the utilization of the *evaluation* (*fitness*) *function*. The evaluation function is a function that provides guidance on the quality of a candidate solution by comparing it to other solutions (e.g., optimal, within population of solutions, etc.). The evaluation function can be derived from the objective function, or it can be a set of some algorithmic (heuristic) procedures that employ additional knowledge gathered during an execution of an algorithm. Because of various ways to construct an evaluation function, one which maps from the space of candidate solutions to a set of real numbers (known as *numeric evaluation function*, [Mic04]) is utilized in this dissertation, and denoted as $ev(\boldsymbol{x})$. Another useful term used in practice is the *solution quality* referring to the evaluation function value of any given candidate solution.

Global optimum can be hard to identify and even harder to locate. Special case is convex programming, where all local solutions are also global or some nonlinear problems where global solutions do not exist [Noc99]. Therefore, definition of the problem provided by set of equations (2.1) may not provide enough information about what kind of optimum we want or are able to find, i.e., *local* or *global*. The notion is fundamentally connected with optimization methods one wants to implement, but also, with the overall number of existing optimal solutions. Formalization of the problem of finding local optimum is connected with the notion of the neighborhood, which needs to be introduced.

Definition 3. Given an optimization problem with instances $(\mathcal{X}, f)$, a *neighborhood* is a mapping

$$\mathcal{N} : \mathcal{X} \to 2^{\mathcal{X}} \tag{2.3}$$

defined for each instance [Pap98]. ◇

Definition 3 is more general than the classical definition of the neighborhoods in the real analysis. It is letting us determine a neighborhood $\mathcal{N}(\boldsymbol{x})$ of some solution $\boldsymbol{x} \in \mathcal{X}$ as a set of feasible solutions that are being generated in its *vicinity*, by applying

some *elementary transformations* on $\boldsymbol{x}$. For example, in TSP, a neighboring solution of the feasible tour $\boldsymbol{x}$ can be obtained by exchanging some two edges. In SAT, two solutions can be neighbors if the difference is in only one bit. Therefore, the size of the neighborhood can be determined in terms of the number of executed transformations that have generated corresponding neighboring solutions.

Definitions of local and global optimum for general form of optimization problem are presented in the following [Pap98, Dav06a]:

Definition 4 (Local optimum)**.** A feasible solution $\boldsymbol{x} \in \mathcal{X}$ is a *local minimum* (*local optimum*), or *locally optimal* with respect to $\mathcal{N}$ if

$$f(\boldsymbol{x}) \leq f(\boldsymbol{y}), \ \ \forall \boldsymbol{y} \in \mathcal{N}(\boldsymbol{x}). \tag{2.4}$$

◇

Definition 5 (Global optimum)**.** A feasible solution $\boldsymbol{x} \in \mathcal{X}$ is a *global minimum* (*global optimum*) of the objective function $f(\boldsymbol{x})$ if

$$\nexists \boldsymbol{y} \in \mathcal{X}, \text{such that} f(\boldsymbol{y}) < f(\boldsymbol{x}). \tag{2.5}$$

◇

Previous definitions can be generalized to a maximization problem, i.e., observing the function $-f(x)$.

The aim of any optimization method is to locate global minimum, or find solution of the *best* quality. However, finding multiple (global and/or local) optimal solutions can be useful, especially in engineering, for example, when locating some hidden properties. In addition, solving optimization problem is usually simplified to finding one optimal solution, even when the problem contains multiple optimal solutions (solutions that correspond to the same minimal or maximal value of the objective function) [Dav06a].

2.1.2.2 Classifications of optimization problems

Optimization problems can be classified in many different ways, according to the[(3)]:

- **Types of variables:** *continuous* and *discrete*;
- **Definition of the feasible space:** *constrained* and *unconstrained*;
- **Number of optimization criteria:** *single-objective* and *multi-objective*;
- **Number of optima:** *uni-modal* and *multi-modal*;
- **Uncertainty:** *deterministic* and *stochastic*.

Classification based on the types of the variables divide optimization problems naturally into three categories: problems with continuous variables, problems with discrete variables (which are also called *combinatorial*) and problems involving both types of variables which are called *mixed*. Combinatorial optimization is formally defined as

[(3)]The classification was inspired by general guidelines provided on http://neos-guide.org/content/optimization-taxonomy

"*the mathematical study of finding an optimal arrangement, grouping, ordering, or selection of discrete objects usually finite in numbers*" [Law76]. In the discrete (combinatorial) problems, we are looking for an object that may only take discrete values, e.g., from a finite or, possibly, countably infinite set, typically an integer. By contrast, in the continuous problems feasible solutions are allowed to take any value restricted by constraints, typically real numbers, or even be a function. Discrete optimization problem is also known as *integer programming problem*. If a model contains continuous and discrete variables, it is referred to as *mixed integer programming* (MIP) problem [Pap98, Noc99, Blu03, Tal09]. A very good survey related to historical development of different combinatorial problems can be found in [Ale05].

Classifications of the optimization problems based on a definition of the feasible space are the most diverse. Unconstrained types of optimization problems have no limitations on values of variables other than they minimize (or maximize) a value of an objective function. They are usually tackled by classical methods, such as the gradient or Newton's method. However, for constrained types of problems the character of constraints is very important since they may vary widely, from simple bounds to complex systems of (in)equalities. Therefore, based on the nature of constraints, optimization problems are further classified as: *convex*, *linear* and *nonlinear*. For example, when both, an objective function and all the constraints, are linear functions the problem is known as *linear programming problem*, whereas *nonlinear* programming problems have at least some of the constraints as a nonlinear function. The nature of the objective function can also greatly influence the diversification of optimization problems. Beside already mentioned linear and nonlinear programming problems, there are some other problems such as *geometric* and *quadratic* programming problems. In geometric programs an objective and constraint functions take a special form, i.e., an objective function is expressed as "posynomial" function (different from "polynomial" as the posynomial exponents are real numbers with strictly positive coefficients) and constraint functions as "monomials" [Boy07]. In quadratic programming an objective function is quadratic function and constraints are affine (linear). In case the smoothness of a constraint functions is considered, optimization problems can be classified as: *differentiable* and *non-differentiable*, which are the topic of calculus of variations [Noc99].

Given the number of objective functions, *single-objective* and *multi-objective* programming problems are distinguished. Owing to the technology advances, operations research grew rapidly while exploring practical problems that need to incorporate a number of contradictory objective functions, in which case we are dealing with multi-objective optimization problems. A problem we can all relate to is the purchase of an automobile car, as we try to minimize the costs while maximizing comfortability. The multi-objective problem can be trivial when solved with single-objective methodologies, i.e., when optimal solution of each objective function can be found. In the case of nontrivial multi-objective problems, between two extremes can exist many other solutions, the so called *trade-off* solutions. All trade-off solutions are optimal solutions to a multi-objective optimization problem and are called *Pareto-optimal* solutions [Deb14].

The optimization problem is *uni-modal* if there is only one unique optimal solution. Otherwise, it is a *multi-modal* optimization problem. Multi-modal problems contain many optima, including at least one global optimum and a number of local ones in the search space. A goal of the multi-modal optimization is to identify as many of these

optima as possible. There is a significant difference between the multi-modal and multi-objective problems. In most multi-objective optimization problems, the Pareto-optimal solutions have certain similarities in their decision variables. However, between one local or global optimal solution and another in a multi-modal optimization problem, such similarity may not exist [Deb14].

The uncertainty is a big part of the problems in everyday life. The absence of uncertainty defines a class of optimization problems called *deterministic optimization problems*. However, great number of real-data can be, mainly, influenced by different kinds of errors (e.g. measurement errors), which might notably impact the correctness of purported results. Most often this errors cannot be removed. Furthermore, uncertainty often happens in many economic and financial planning models while working with some data information that cannot be specified precisely. Problems that have incorporated uncertainty belong to the class of *stochastic optimization problems*.

We define an *instance* of an optimization problem as a pair of the corresponding feasible region and objective function, i.e., $(\mathcal{X}, f)$. For example, the instance of TSP contains number of cities and the matrix of distances between them. An optimization problem can, thus, be represented as a set of instances for which we are given some input data and enough information to obtain a solution [Pap98].

2.2 Optimization methods

Optimization method represents a search procedure for obtaining an optimal solution of an optimization problem, possibly subject of some set of constraints. Various types of optimization methods have been developed over the years for dealing with hard optimization problems. Therefore, we begin this section with a historical background of their development. As previosly, there are different ways to distinguish optimization methods. For example, some are constructed to find only local optima, while others try to find a global optimum. A thorough scheme of optimization methods classification may be found in [Hau04].

2.2.1 Background

The techniques for solving optimization problems can be traced back to the era of Pythagoras (around 500 BC). Beside the logical interpretations within mathematics of that age, one can identify a pursuit of scientists towards formalism in order to capture an *optimal* state of some observed natural or other process. For example, Pappus of Alexandria (300 BC) was concerned with a question of honeycomb shape, and proved the so called *honeycomb conjecture*, that repeating hexagonal pattern represents an optimal way of storing honey [Kir14]. However, to proceed with a construction of some general optimization methods, first a language of algebra and calculus had to be established. From the work of Al-Karaji in 11th century, who introduced the theory of algebraic calculus, around 500 years has passed before René Descartes proposed mathematical principles and notation that have enabled description of the geometric objects by using language of algebra, thus opening the doors for later concepts such as *calculus of variations*. The calculus of variations is considered to be a key stone for development of optimization, and later on, operations research, as it offers methods

that deal with finding maximum or minimum of functionals. Methods were usually constructed as procedures of finding an extreme point of some empirical or theoretical function. Karush-Kuhn-Tucker conditions, maximum principle and Hamilton-Jacobi-Bellman equation were among popular methods for solving optimization problems at the beginning of the 20's century [Gal91].

The modern optimization methods were developed mostly with a help of the *Operations research (operational research)*, abbreviated OR. The origin of the discipline can be traced to the work of Charles Babbage (1791-1871), who already worked on problems of sorting, pricing and transportation operations in the postal industry at the beginning of the 19's century. His paper "On the Economy of Machinery and Manufactures" from 1832 was an influential early work of operations research, however, operations research officially started to exist after 1940's and the second world war. The earliest occurrence of the expression is from 1941, given by British scientist Patric Blackett. In the report of the US navy office, called *Methods of operations research*, the meaning of the phrase "operations research", or British "operational", was regarded as a study of warfare, i.e., operations of war which were under the control of military departments [Mor46]. Soon after the end of the second world war it was noticed how broad an application of the operations research has became, since it was involving work from many different aspects of science. The importance was recognized both in theoretical and real-life problems, for example, in the field of management in terms of helping businesses to achieve their goals.

It can be stated that the development of modern optimization methods happened on the crossroad of operations research and mathematical programming[(4)]. In parallel with operations research, mathematical programming was developed. Leonid Kantorovich (1912-1986) was one of the founders of the *linear programming*, a mathematical formulation of the problem represented by linear equations and/or inequalities. He was the first to formulate a problem of finding the best (optimal) solution under certain restrictions, so called *Problem C*, and has presented a new method for solving it, today known as *simplex method*. The phrase was coined later by the influence of George Dantzig, in 1948. The widely used simplex algorithm was based on the idea of improving the value of objective function by moving from vertex to vertex of a polytope. After 30 years of refinement, today *simplex algorithm* is considered as very efficient, with hundreds of variables and thousands of constraints being solved routinely [Pap98, Ale05]. The concept of *dynamic programming* by Richard Bellman came in 1940. Very thorough exploration of some aspects of the operations research development before 1960's is provided in book of Schrijver [Sch98] and other good books and proceedings [Ale05, Joh06, Grö12, Kir14].

2.2.2 Classification

The choice of a method or the construction of a new one depends to a large extent on characteristics of the problem to be solved. As stated in Chapter 1, we are primarily focused on combinatorial search problems, although many of our conclusions can be extended to continuous optimization problems.

Methods for combinatorial optimization problems are in the most general sense divided into *exact* (*complete*) and *approximate* (*incomplete*) [Dav06a, Tal09]. The ap-

[(4)] The expression *mathematical programming* is used as a synonym for optimization [Der09].

proximate methods represent method such as: (*i*) heuristic, (*ii*) approximation, and (*iii*) meta-heuristic methods. In the literature one can find different classifications. In [Hau04] six categories of optimization methods are distinguished, while [Tal09] provides a taxonomy in greater detail.

Exact methods (algorithms) are based on the search of the entire search space and guarantee to find an optimal solution in bounded time for every finite size instance of a combinatorial problem. The most typical example of implementation of such methods is through *exhaustive search*. Exhaustive search is a technique for checking every solution in the search space, after which the best-found solution designates the global optimum. The exhaustive search methods are also known as *enumerative algorithms* [Mic04]. However, for combinatorial problems that are NP-hard no polynomial time algorithm has been found. In other words, the main problem with solving combinatorial problems is that the number of feasible solutions can be extremely large or even infinite. Often this makes exact algorithms too expensive in computational time. This invites for other intelligent ways to guide the search towards more promising parts of the search space in order to avoid useless examinations. Such methods are, for example: branch and bound B&B, branch-and-cut B&C and branch-and-price B&P [Blu03, Sta07, Tal09]. Alternative approach for solving combinatorial problem would be to use other formulations, such as linear programming, to describe the original problem. If problem can be presented in a way that feasible solutions correspond to vertices of some convex polytope, the linear programming methods might be employed. Efficient exact algorithms for linear programming problems are a well known simplex method or interior point methods [Law76, Tal09].

Heuristic methods are, for the most part, developed when utilization of the exact methods becomes cumbersome or even impossible due to large dimensions of optimization problems. Heuristics are techniques for solving complex optimization problems that employ *a priori* knowledge about the considered problem. The term *heuristic* originates from Latin, meaning *to find* or *to discover* using trials and errors. Today it is used in its original form, pertaining to the trial-and-error method of problem solving. Their main characteristic is to provide solutions of relatively high quality, "close" to the optimal or *near-optimal*, in reasonable amount of time. Most common used technique in heuristic methods relies on local information around some starting feasible solution $\boldsymbol{x}$. Such techniques are known as *local search strategies*, LS. The main idea behind LS is that during the search for the nearest optimum a sequence of good feasible solutions is generated by conducting exploration of $\mathcal{N}(\boldsymbol{x})$. Neighborhood exploration represents a sequence of consecutive execution steps, each leading to a better solution. The weakness of the LS strategies is that, in general, they greatly depend on the size of the neighborhood and the selection of the starting point, leading the search usually towards a local optimum. Nevertheless, if the size of the neighborhood is large enough then the local technique decisions may lead the search towards the global optimum. LS methods are typical representatives of so-called *iterative heuristics*.

Heuristic methods, in general, do not guarantee the quality of the obtained solution. However, there are other approximate methods designed to make such guarantee, such as *approximation algorithms*. The methods are founded on the factor of the approximation, α, which is utilized to establish the distance of the worst generated solution from the optimum. Approximation algorithms are deterministic algorithms that guarantee that the obtained solution will be at most α times worse than the optimal, for any

problem instance. The property does not imply that the algorithm will not generate an optimal solution, rather, the obtained result will never be worse than optimum for the specified factor α.

Contrary to the heuristics, which were designed for solving specific problems, meta-heuristics (also known as *black-box* methods) are general (universal) computational methods designed to deal with various optimization problems. They iteratively generate and/or improve solutions by applying some predefined stochastic rules. The generality implies that they do not use *a priori* knowledge about the problem to be optimized. As such, meta-heuristics might invoke any heuristic method in order to guide them in the search for the best solution of a given particular problem. The main goals of each meta-heuristic method are: solving problems faster, solving large problems, and obtaining robust algorithms. They accomplish these goals by efficiently exploring suitably selected sub-spaces of a very large solution space for each given problem. Numerous meta-heuristic methods have been developed in the past twenty years. Most of them are known to provide solutions of high quality for a reasonable execution time.

2.2.3 Heuristic methods

Today's technology allows us to search for solutions despite of increasing complexity of demands. Nevertheless, solving real-life problems may be difficult since the number of factors that need to be accounted has been increasing in the last decades. In addition, the restrictions under which we deal with problems in the real world may be different from those required by the classical solution methods such as linear, nonlinear and dynamic programming. Namely, it is quite common that the results are required in a timely manner, i.e., the demands are mostly related to finding high quality solutions as fast as possible. One can state that getting any solution fast is much better than having to wait for superior one. However, obtaining high quality solutions as fast as possible may create various mistakes. The most abundant one is changing the purpose of the problem so it would fit the existing algorithm.

2.2.3.1 Type of solutions

During the optimization process we distinguish *partial* from *complete* solutions. A partial solution may appear as [Mic04]: (*i*) incomplete, or (*ii*) a complete solution of a reduced (simpler) problem. An incomplete solution is used when we focus our attention on a subset of a search space that has a particular property. Furthermore, a partial solution is incomplete solution if, e.g., some of decision variables are not assigned with fixed values. For example, an incomplete solution is the one where we have fixed the values of some decision variables, and still are left to determine the values of the rest of them. Good example is for TSP of n cities where an initial partial solution is presented as a solution of some k cities ($k < n$), so the incomplete solution has $(n - k)!$ different states (Fig. 2.1). To find a complete solution we need to search the subspace of the search space of size $(n - k)!$. The notion of the incomplete solution is important for methods that use ordering as a part of their search technique [Mic04].

To attain a good final solution of an optimization problem we could decompose the problem into a set of simpler (smaller) problems. The complete solution of the smaller problem is then considered as a partial solution of the original problem. Combining

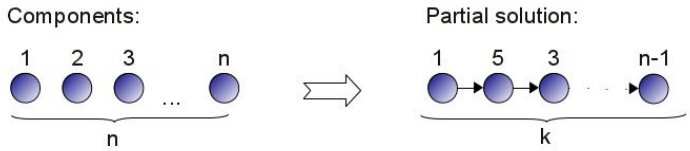


Figure 2.1: Partial solution for TSP

solutions of the considered sub-problems might lead to either high quality solution or an optimal solution of the original problem. This can be demonstrated on TSP with n cities so that we are searching for shortest paths in smaller groups of the cities, at the end combining the found paths into a final complete tour [Mic04].

2.2.3.2 Classification of heuristics

There are different ways to classify heuristics. We point out the classification based on how a heuristic is treating a solution and distinguish two broad classes of heuristic methods: (*1*) **constructive** and (*2*) **improvement**. Both classes of algorithms provide at the end a *complete* solution, i.e., all the decision variables are specified [Mic04].

The *constructive heuristics* are building the solution either from some empty set or an initial partial solution, peace by peace iteratively adding appropriate solution components. These heuristics are usually the fastest methods. They cannot be interrupted at any time since it might result with an incomplete solution, which is not valid. Typical representative of the constructive class of algorithms is the *greedy method* [Tal09]. In particular, a variable (component) is assigned with a value in a best possible way regarding the current state of the search. Heuristics that implement greedy methods are known as *greedy heuristics*.

On the other hand, *improvement techniques* start from a complete initial solution and try to find a better solution by applying some transformations. They are evaluating only complete solutions, therefore, we are allowed to interrupt them at any time. Improvement strategies are an essential part of a LS algorithms, which is why *improvement heuristics* belong to the class of local search methods. Detailed classification of heuristics is provided in [Dav06a, pg. 33].

2.3 Examples of optimization problems and their methodologies

In this section we present two well-know combinatorial optimization problems: scheduling and satisfiability problems. For each we provide a brief general overview and classifications. We then continue by focusing on their specific variants, formulations and established heuristic methods. We are especially interested in the heuristic methods employed in this dissertation within development phases of BCO algorithms.

2.3.1 Scheduling problems

Various types of scheduling problems are recognized in many scientific disciplines and everyday life. The most common scheduling problems from the real life are known as *sequencing problems* such as: customers standing in a queue, programs to be executed on a computer server, aircraft waiting for take-off clearance, etc. Generally, scheduling problems represent a class of combinatorial optimization problems which deal with allocation of tasks to several resources [Pap98, pg. 310]. Among the first attempts to organize research on scheduling problems is conducted in book of Conway, Maxwell and Miller in [Con67]. Of course, great deal of research about scheduling problems has been conducted in already developed field of scheduling. For example, parallel machine scheduling is studied in the late 1950s addressing the problem of scheduling one-stage tasks on several machines to minimize the total loss, which is a sum of losses associated with the individual tasks [McN59]. In [Bel56] Belmman addresses scheduling problems in the context of a control (management) and transportation and reviews current techniques for solving such problems.

The conducted studies lacked the general overview of the field of scheduling and, as a result, Conway, Maxwell and Miller sought for the model under which scheduling (sequencing) problems may be generally described, in order to encompass sequencing problems dealt within the literature of that time. They presented their overview in [Con67] claiming that it is not possible to generate a mathematical model over the entire process of making a decision, and that proposed abstract model does not represent a solution to any real sequencing problem. In their book they suggest that the model might be utilized as an information along with the common sense and knowledge about other aspects of the considered scheduling problem. A term *job-shop scheduling problem* was coined. The basic unit of the job-shop process is introduced as *the operation*, where the definition of a *job* assumes that jobs are a subset of the set of all operations that can be disjoint, exhaustive or mutually exclusive. A *machine* is defined as a device capable of performing any operation, but abstractly, should refer to a time scale with certain intervals available. A *job-shop* can then be observed as a set of all the machines that are identified with a particular set of operations. The proposed abstract model can thus be referred to as scheduling a *job-shop process* which consists of the machines, the jobs (operations) and a statement of the disciplines that restrict the manner in which operations can be assigned to specific points on the time scale of the appropriate machine. In other words, scheduling a job-shop process is the task of assigning each operation to a specific position on the time scale of the specified machine [Con67].

Job-shop problems do not offer complete overview of all scheduling problems that are considered today, therefore, further elaboration about basic notation is required in order to provide general definition. Typically, a basic unit used to model activities to be scheduled is in the literature most often referred to as a *task* [Bła07, pg.57]. Moreover, scheduling problems can be defined by three sets: set of n tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$, set of m machines (machines) $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ and set of s types of additional resources $\mathcal{R} = \{R_1, R_2, \ldots, R_s\}$ [Bła07]. In particular, a tasks $T_i \in \mathcal{T}$ can be characterized by [Len77, Bła07]:

- *Processing time* p_{ij}, $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots m$, is a time needed to process task T_i by machine P_j.
- *Arrival time* or *release date* r_i, which is the earliest possible starting time at which

tasks T_i is ready for processing.

- *Due date* d_i referring to time limit by which task T_i should be completed, and *deadline* $\tilde{d}_i$, specifying "hard" real time limit.
- *Weight* w_i denoting relative urgency of task T_i.
- *Resource request*, when specified, is classified into different types and categories [Bła07, pg. 425].

A *job* can now be defined as a one of the n subsets which tasks can shape [Bła07]. Based on this introduction it is clear that scheduling problems are versatile and their classification can be organized by different set of assumptions, which is more clearly elaborated in the next section.

2.3.1.1 Classification of scheduling problems

Classification upon which we distinguish different categories of scheduling problems may be conducted in various ways. For example, scheduling problems can be distinguished by a different set of assumptions, of which most common are: number of tasks to be processed, the order in which the tasks are processed on machines and the manner in which tasks are arriving [Con67]. Another general classification in the literature is based on the amount and type of information provided for tasks (jobs) before or after a scheduling starts, where we distinguish two categories [Dav06a]. When all information concerning the scheduling of tasks is known either *a priori* or gathered during the running time, we deal with *deterministic scheduling*. If necessary information cannot be provided or we need to use some presumptions regarding, for example, the number of tasks (jobs), their due dates, or the processing times, then we are dealing with *stochastic scheduling* due to the presence of the uncertainty [Pin04, Dav06a]. Another common classification is oriented towards specific times of release of information, where two categories are distinguished: *dynamic* and *static scheduling* [Con67, Dav06a, Sin07]. In the static scheduling all information is always available *a priori*, while in the dynamic scheduling the information is provided during running time. Often there is a need to know when information is being collected, yielding another arrangement of scheduling problems. The collection of information can be conducted *offline* and *online*. Hence, in *offline scheduling* all information is collected before the scheduling starts and is often associated with static scheduling. In *online scheduling*, some or all information is collected during running time. The later is often referred to as *dynamic scheduling*. However, we can utilize dynamic scheduling without disclosing all information before scheduling starts [Bes04].

To supplement general overview, we should note that a taxonomy based on the four-parameter notation for identifying scheduling problems was first time offered in [Con67]. The notation is presented with four capital letters A/B/C/D and was later modified to n, m, l, k in [Len77]. A thorough taxonomy covers job-shop and flow-shop problems and is too extensive to cover here. Instead, we review classifications based on assumptions that are common in today's literature. The forthcoming survey is inspired by a commendable introduction provided in [Bła07, pp. 57-58]. Scheduling problems can be classified by the specialization of the machine employed, i.e., we distinguish *parallel* and *dedicated* machines. Parallel machines perform the same functions, whilst dedicated machines that are often specialized. Furthermore, parallel machines can be

identified as: (a) *identical*, when all machines have the same speed, (b) *uniform*, if each machine has different speed that does not depend on the task, and (c) *unrelated* if machines have different speeds that depend on the tasks. Unrelated machines can also be further classified as: (a) *flow shop*, (b) *open shop*, and (c) *job shop*. A great deal on job shop was already mentioned. The flow shop is a case where all jobs have prerequisites (routes) in the order in which they will be processed [Con67]. It presumes that all machines will process all jobs. Open shop is similar to flow shop, however, the order of tasks comprising one job may be arbitrary [Bła07, pg. 321].

The classification of scheduling problems used in today's literature was proposed by Graham in [Gra79] and Błażewicz in [Bła83]. A notation $\alpha|\beta|\gamma$ is commonly utilized to characterize a scheduling problem with α, β and γ denoting the machine environment, the scheduling characteristics and restrictions, and the scheduling objective criterion, respectively. The first parameter α refers to the type of machine, like single ($\alpha = 1$) or parallel identical machines ($\alpha = P_m$). Moreover, the machines can have different speeds, in which case $\alpha = Q_m$ or can be unrelated ($\alpha = R_m$). Index m indicates that the number of machines is fixed, but it can also indicate fixed number of stages in which case letter s is used. Parameter β is specifying job constraints like dependencies between the tasks ($\beta = \texttt{prec}$) and the exhaustive list of all its possible scenarios is given in [Che99] and [All08]. In order to provide few examples, another useful terminology regarding scheduling problems is related to the interruption of the currently executed task. If the intention is to interrupt the task in order to resume its computation later, then the act is known as *preemption*. *Non-preemptive* scheduling doesn't allow interruptions [Bru09]. If preemptions are allowed the field β includes `pmtn`. Other cases concern specifying additional resources, processing times, deadlines, etc. Parameter γ describes the objective to be minimized, like makespan, total completion time, maximum lateness, total tardiness, release dates, etc. To elaborate on one example let completion times of tasks $T_1, \ldots, T_n$ be $C_1, \ldots, C_n$. A minimization of the objective function $C_{max} = max_i\{C_i\}$, (where C_{max} is known as *makespan*) referrers to finding minimal completion time of the last job to leave the system. For example, $P||C_{max}$ is a problem of scheduling independent jobs on identical machines to minimize makespan without preemptions. In [Dav06b, Dav06a] Multimachine Scheduling Problem with Communications Delays (MSPCD) was considered and is classified as $P^*|\texttt{prec}^*|C_{max}$. The term P^* is used to denote that machines are not completely connected, interconnection topology is given as an input parameter instead. In addition to preceding relation among tasks (`prec`), the time required to transfer data between dependent tasks is considered. Therefore, the value of β field is set to $\texttt{prec}^*$. The communication time depends on the distance between machines allocated to the corresponding tasks.

The topic of the dissertation is a problem that belongs to the class of scheduling problems known as *multiprocessor scheduling* problem. Its general definition is provided in [Gar79, pg. 238] as a problem of finding minimum possible time required to schedule all n jobs to m machines. Restricting the problem with different set of assumptions produces a vast number of multiprocessor scheduling problems. We were interested in one particular, that is $P||C_{max}$, described in the forthcoming text.

2.3.1.2 $P||C_{max}$

In this thesis we address a problem of *static scheduling of independent tasks on identical machines* (*homogeneous multiprocessor systems*), i.e., $P||C_{max}$. The expression *static*

indicates that the total number of tasks is known, as is the duration of each task. In addition, the multiprocessor system contains m identical machines where the solution of the considered scheduling problem consists of the index of the associate machine and starting time for each task. The objective of the problem is to find minimal *makespan*, which is known to be NP-hard in a strong sense [Kar72, Gar79, Gla94, FP10]. Makespan minimization on parallel machines is a fundamental and extensively studied scheduling problem and is important in practice since the resourceful use of multiprocessor systems depends highly on adequate schedule of tasks on machines [Dav09, Alb12]. The research on this problem dates back to 1960s according to [Mar73].

The interest in the study of scheduling problems on parallel machines arose as a result of interest in the usage of computers with many processors in parallel [Gra66, Dav06a, Fra10, Pin12]. However, scheduling problem exists everywhere. It is a part of a great number of practical problems which are usually complex, like bandwitdh scheduling, airport gate scheduling, machines in a workshop, repair crew scheduling, agriculture, hospitals, transport, virtualized environments, many-core processors, DNA sequencing [Rob09, Fra10]. In the literature many scheduling problems are addressed, but still real world problems can use only a fraction of provided solutions when faced with new challenges. One example of such real problem is presented in paper [Boc09] in the context of shoe manufacturing. The problem is identified with a parallel machine scheduling problem (PMSP). The novelty in the scheduling of operations on a multiprocessor machine in the context of shoe manufacturing is that the machines are not independent, that is, all the processors share some data and whenever a single processor requires a setup the entire machine should be stopped.

2.3.1.3 Complexity and problem formulation

Solving $P||C_{max}$ to optimality, in its most general form, belongs to the class of NP-hard problems, as proved in the book by Garey and Johnson [Gar79]. In few cases solution can be found in polynomial time, such as when all tasks have unit length [Dav06a]. The simplest case of NP-hard is for $m = 2$, while for arbitrary m scheduling problem becomes NP-hard in a strong sense [Gar79].

The first mathematical formulation of $P||C_{max}$ was presented in 1994 by Queyranne *et al.* [Que94] in the form of integer program. However, we use formulation by [Mok04], described in the following.

Let m be the total number of identical machines engaged, and n number of non-preemptive tasks. The considered scheduling problem consists of assigning tasks to processors, as well determining their starting times. Let $T = \{1, 2, \ldots, n\}$ be a given set of independent tasks, where each task $i \in T$ has to be processed by exactly one out of the set of identical machines $P = \{1, 2, \ldots, m\}$. Each machine can engage only one task at a time, and once the task has started it will continue to run on the same machine until completion. Let l_i be the processing time of task i ($i = 1, 2, \ldots, n$), which is known and fixed, and y_j ($j = 1, 2, \ldots, m$) the load of machine j calculated as the sum of processing times of all tasks assigned to machine j. The goal is to find a schedule of tasks on processors such that the corresponding completion time of all tasks $max_{j \in P} y_j$ is minimized [Dav12].

A solution of a scheduling problem might be graphically represented by Gantt diagram. In Fig. 2.2 an example of the schedule for 9 tasks on 4 machines is shown [Dav12]. The horizontal axis in Fig. 2.2 represents time, while the machines are enu-

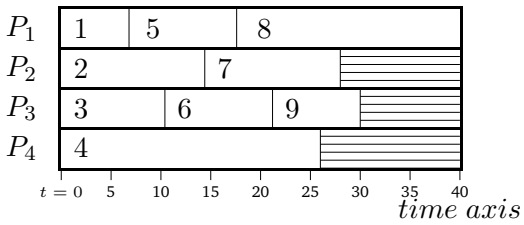


Figure 2.2: Gantt diagram–schedule of tasks on machines [Dav09, Dav12]

merated on the vertical axis and the rectangles represent tasks. To determine the starting time of a task one should sum up the processing times of all tasks already scheduled to the same machine. The total completion time for the schedule shown in Fig. 2.2 equals 40 time units (the completion time of task 8 scheduled on machine P_1). This schedule is obtained by sorting tasks in the increasing order of task indices and scheduling them to the least loaded machine. Any schedule that has a completion time less than 40 time units is considered better. The goal is to discover the schedule of tasks on machines that has the shortest completion time.

In order to present a mathematical programming formulation of the problem, let us introduce the binary variables x_{ij} defined in the following way:

$$x_{ij} = \begin{cases} 1, & \text{if task } i \text{ is assigned to machine } j, \\ 0, & \text{otherwise.} \end{cases}$$

The considered scheduling problem is formulated in the following way [Mok04]:

$$\min y = \min \max_{j \in P} y_j \tag{2.6}$$

$$s.t. \quad \sum_{j=1}^{m} x_{ij} = 1, \;\; 1 \leq i \leq n, \tag{2.7}$$

$$y - \sum_{i=1}^{n} l_i x_{ij} \geq 0, \;\; 1 \leq j \leq m, \tag{2.8}$$

$$x_{ij} \in \{0, 1\}, \;\; 1 \leq i \leq n, \;\; 1 \leq j \leq m, \tag{2.9}$$

$$y \in \mathbb{R}^+. \tag{2.10}$$

The objective function that should be minimized represents the total completion time of all tasks - makespan y. Each task i should be scheduled to one and only one potential machine j (constraints (2.7)). The makespan is computed as the maximum over all machine's completion times, and the completion time of a machine is defined as the sum of processing times of all tasks scheduled to that machine. This is described by the constraints (2.8). Constraints (2.9) show the binary nature of the variables x_{ij}. The objective function value y is a positive real number as it is stated by (2.10).

2.3.2 Methods for $P||C_{max}$

Exact algorithms, utilized to solve to optimality small to medium size problems, represent one line of research in the theory of scheduling [Del12]. The most common enumerative algorithm is the branch-and-bound algorithm (B&B). The B&B aims to find

appropriate bounding scheme and dominance rules that restrict the search by eliminating partial solutions [Del95]. Dominance rules and bounding schemes are usually based on problem-specific features [Che99]. [Mok04] successfully developes an exact cutting plane algorithm to address the problems with up to 1000 tasks and 100 processors. Another line of research are approximate methods designed to search for high quality solutions for a broader class of scheduling problems and/or when enumerative algorithms fail to produce results.

The first deterministic online algorithm for multiprocessor scheduling problems was presented by Graham in 1966 in [Gra66]. The author has introduced a simple greedy heuristic method, known as *List scheduling* (LS) which today represents a whole class of algorithms. LS constitutes of two steps: (1) ordering of jobs by some given *priority rule* (e.g., second job from the list cannot be considered for scheduling before the first from the list), and (2) scheduling the jobs on machines following some specified criterion. For example, current task is allocated to an idle processor or to processor that will be the first to complete its execution. The second step is in the literature referred to as *Earliest-start* strategy, ES. The worst-case performance guarantees for the LS algorithm are presented in [Gra66]. Other variants of the first step of LS are discussed in [Gra69]. A simple rule, today known as *Largest Processing Time first* rule (LPT), implies sorting the jobs in nonincreasing order of processing times before they are scheduled to the least loaded machine [Che04, Dav09]. Implementing this simple rule in the LS algorithm Graham has obtained performance guarantee through theoretical analysis and has obtained the upper bound of $4/3 - 1/(3m)$ as the worst-case scenario. Numerous approximate algorithms were proposed since then. Generally, the algorithms with stronger guarantees are preferred [Che99].

Based on the fact that the $P||C_{max}$ problem can be formulated as a *bin-packing* problem (BPP), Coffman, Garey and Johson proposed *MultiFit* algorithm [Cof78, Che04, Dav09]. Bin-packing problem is a problem of packing a given set $I = \{1, 2, \ldots, n\}$ of n items into as few bins of a fixed capacity C as possible. The principle of duality between $P||C_{max}$ and BPP was considered by many authors [Che04, Dav09]. Since the development of meta-heuristics their application in solving various scheduling problem has been successful [Osm96b, The98, Che99, Leu04, Tal09]. A survey of the first implementations of meta-heuristics dealing with parallel machines scheduling problem may be found in [Osm96a, FP10].

2.3.3 Satisfiability problem

In this section we review definition of a satisfiability problem and give a short survey on the most common corresponding solvers in order to locate methods used as underlying mechanisms of BCO for 3-SAT.

A topic of our research belongs to a class of *propositional satisfiability problems*, abbreviated as SAT. The historical significance of SAT originates from the proof of its NP-completeness. Moreover, SAT is crucial in computation complexity theory as it provides basis for determining complexity of other algorithms. Aiming to solve SAT problems represents a quest for solving other problems as well. For example, SAT can be used in bioinformatics [Lyn06], AI planning, software model checking, etc. [MS08]. It represents one of the most studied problems in the field of artificial intelligence and operations research.

In the field of theoretical computer science a *decision problem* can be viewed as a form of posing a question, usually formulated with 'if ...' or 'if true ...', with only two possible answers: YES or NO. For example, for two given numbers x and y, decision problem within framework of computability theory can be formulated as a question of divisibility of two numbers. An answer can be either YES or NO, dependent on the values of x and y [Ogn04]. Moreover, a problem is *decidable* if the answer can always be provided inside some formal framework. The formalism behind expressions of decision problems is studied in the field of formal logic. Specifically, areas such as propositional logic provide rules of syntax and semantics that help mapping these expressions to a system of truth values [Hoo98a].

Rules of propositional logic describe construction of *propositional formulas*. Propositional formulas assume basic building blocks such as: *propositional variables* and *logical operators*, for example, conjunction ($\wedge$) and disjunction ($\vee$). Propositional formulas are used to articulate different statements that can have unique truth value (TRUE or FALSE) and have become essential in many fields of science such as mathematics and philosophy. Occasionally a propositional formula might be needlessly too complex, especially if other forms of expressions are more suitable, such as *normal formula*. For instance, a propositional formula $((C \wedge D) \vee A)) \wedge ((C \wedge D) \vee B)) \wedge (E \vee \neg E)$ is equivalent to the normal formula $(C \wedge D) \vee (A \wedge B)$ [Hed04, pg. 11]. Building blocks of normal formulas are *literals* which can be propositional variables or their negation. Normal formula, represented as a conjunction over disjunctions of its literals, is known as *conjunctive normal form* (CNF). The disjunctions are called *clauses*. In a general form, the CNF formulas with k literals (L_{i_j}) are all the formulas F defined over set of n variables X and m clauses (C_i), for which:

$$\begin{aligned} &F = f(X) = C_1 \wedge C_2 \wedge ... \wedge C_m, \\ &X = \{x_1, x_2, x_3, ..., x_n\}, \\ &C_i = L_{i_1} \vee L_{i_2} \vee \ldots \vee L_{i_k}, i = 1, 2, ..., m \\ &L_{i_j} \in \{x_l, \neg x_l\}, j = 1, 2, \ldots, k, l = 1, 2, ..., n. \end{aligned} \tag{2.11}$$

Normal formula that is expressed as a disjunction of clauses, where clauses represent the conjunctions, is known as *disjunctive normal form* (DNF).

SAT in propositional logic is a decision-making problem in which the goal is to determine whether formula (2.11) is satisfiable, i.e., if there exists such set of values that, after assignment to variables, formula evaluates to true [Ogn04]. If a proper set of values does not exist, formula F is called *unsatisfiable*. SAT considers Boolean variables, whereas clauses impose certain restrictions, e.g., it might not be allowed to simultaneously assign x_1 to true, x_2 to false, and x_3 to false.

k-SAT represents special case of SAT problem, namely, when clauses consist of only k literals. For example, a CNF formula of a 3-SAT with four variables can be written as:

$$F = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_4).$$

Originally, SAT is formulated as a decision problem. We may distinguish two approaches to SAT: *model-finding* and *theorem-proving* [Sel92]. Model-finding problem is a problem of finding a satisfiable assignment to a given formula. Within theorem-proving we search for a formal proof that the formula in question is satisfiable. More often in the literature the authors are interested in the *maximum satisfiability problem*

or MAX-SAT, where the objective is to find a model that maximizes the number of satisfied clauses. MAX-SAT is considered as an optimization variant of SAT [Han90, Stü01]. According to [Stü01, pg. 2] the difference in the hardness of MAX-2-SAT w.r.t. the corresponding SAT problem is that *2-SAT problem is a polynomially solvable special case of SAT. Nevertheless, the MAX-2-SAT problem is known to be NP-hard*.

2.3.4 k-SAT solvers

Algorithms for dealing with SAT are employed in different fields, from software verification to computational biology. To illustrate practicality of SAT solvers we point out the construction of logical circuits in the car industry. Thanks to their development SAT solvers are able to deal with many hard satisfiability problems.

In the last decade a lot of sophisticated SAT solvers have emerged, a large number developed as a result of the SAT competition [(5)]. The complexity of new algorithms has increased, along with the number of algorithms' parameters compared to older SAT solvers [Bal09]. Analogous to previously mentioned approaches, two general goals of SAT solvers are identified: confirming that a formula is satisfiable and searching for satisfiable assignment. As a result, we distinguish the *Conflict Driven Clause Learning* (CDCL) from *Stochastic Local Search* (SLS) [Bal14b]. Namely, the general classification of SAT solvers differs *complete* from *incomplete* algorithms [Hoo05, pg. 33]. The complete search algorithms are know as *systematic search algorithms* as they investigate a search space in a systematic manner, able to determine if the solution does not exists. The brute-force approach, where all 2^n assignments are evaluated, guarantees that the solution will be found but disregards the effort needed to reach it. Complete solvers are used in practice to rule out unsatisfiable instances. Majority of complete solvers today are founded on the Davis-Putnam-Logemann-Loveland (DPLL) algorithm [Dav60]. Weakness of complete solvers is their inefficiency while dealing with problem instances for which clause-to-variable ratio becomes higher (until reaching a certain threshold) [Mit92]. To increase efficiency of the search, incomplete solvers are employed.

SLS solvers, also known as model-finding algorithms, are recognized for their speed to obtain solutions of satisfiable instances, especially the hard ones [Hoo98a, pg. 73]. The efficiency of SLS solvers was recognized early in the 1990's when the general problem MAX-SAT was tackled by incomplete solvers, such as SA, LS and and other random search techniques [Han90, Sel92, Sel96]. Because most of the SLS solvers start with a random assignment the main goal is to direct the process of guessing variable values that would lead to a solution. The manner in which variable is chosen depicts each heuristic. A SLS algorithm performs iterative search steps that generally modify a value of the most one variable appearing in the formula, whereas a move is called a *variable flip* [Hoo98a, pg. 73]. After the evaluation of the initial assignment, if all clauses are satisfied solver's work is done. Otherwise, one of the variables is selected and flipped and all the corresponding data structures are updated in order for a new cycle of evaluation to begin. The choice of variable that will change its value depicts each heuristic.

Until today, hundreds of SLS solvers were developed on the top of previous versions, improved by incorporating different data structures and other implementations tricks.

[(5)] www.satcompetition.org

Among vast number of them, general algorithms can be identified that are still the basic part of the most efficient SAT solvers. Of a particular interest for this thesis are two representatives of families of algorithms: GSAT and WalkSAT [Sel94]. The simplest SLS algorithms are considered to follow an uniform random walk paradigm, i.e., choose at random some variable and then change (flip) its current value.

2.3.4.1 Random Walk

A more elaborated version of simples SLS algorithm implements what is known as *conflict-directed random walk* steps [Hoo05, pg. 269]. The algorithm is based on selecting uniformly at random an unsatisfied clause, followed by a random selection of a variable from this clause. The approach is also known as *Focused Random Walk* (special case of *iterative repair* algorithms, introduced later in [Coo97]), targeting a specific type of clauses. Theoretical proof that the focused SLS solvers can solve 2-CNF problems in $\mathcal{O}(n^2)$ is provided by Papadimitriou [Pap91a]. Papadimitriou described the procedure as: *start with any truth assignment; while there are unsatisfied clauses, pick one and flip a random literal in it* [Pap91a, pg. 168], and suggested that the procedure may be used for 3-SAT problems. However, a theoretical proof that a uniform random walk procedure finds an optimal solution of the satisfiable k-CNF ($k > 2$) formulas is presented in [Sch99]. The Schönings's algorithm requires restarts after every $3n$ steps, as shown in Figure 2.3..

Input: Formula F in k-CNF with n variables.
Uniformly at random pick an initial assignment $a \in \{0,1\}^n$.
Repeat $3n$ *times:*
(1) If the formula F is satisfied:
stop and accept the solution.
(2) Let C be a clause (randomly picked) not satisfied by a.
(3) Randomly choose literal $\leq k$ in clause C and flip its value.

Figure 2.3: Main procedure of Schöning's algorithm.

The theoretical work done to explore the efficiency of solvers is usually presented in simple case of random walk, and the theoretical results show that such algorithms provide good results. For that reason random walk is considered as a candidate heuristic for the BCO model.

2.3.4.2 GSAT

The oldest SLS solver, GSAT [Sel92], implements greedy local search techniques. It starts with an initial assignment generated randomly and then changes the value of a variable that produces the largest decrease in the total number of unsatisfied clauses [Sel92]. The flips are repeated until either maximum number of flips is satisfied or a satisfying assignment is found. GSAT requires adjustment of two parameters: number of flips the algorithm performs before it restarts (*MAX-FLIPS*) and number of times the search is allowed to restart before termination (*MAX-TRIES*). Initial recommendations for the adjustment of these parameters are given in [Sel92]. However, to achieve the

high performance on different classes of SAT instances it is essential to conduct a fine-tuning. The weakness of GSAT technique, know to be easily trapped in a local optimum, was reported in[Sel92]. To transcend the obstacle, "sideways" moves were proposed that occasionally allow a variable flip that does not decrease number of unsatisfiable clauses (or, does not increase the number of satisfiable clauses) [Sel92]. If such procedure gets stuck in the local optimum, a simple restart of the greedy search with a new initial assignment is used. Two years later, the same authors proposed three strategies to escape a local optimum: SA, mixed random walk and random noise. Incorporating random walk, focused on the unsatisfied clauses, is a more efficient way for GSAT to avoid being trapped in the local optimum [Sel94, pg. 338]. In the same article a new algorithm, the *mixed random walk* strategy (abbreviated as GWSAT) is proposed. GWSAT mixes a focused random walk strategy with the greedy search, (pseudo-code Fig. 2.4).

```
With probability p, pick a variable occuring in some
   unsatisfied clause and flip its value.
With probability 1 − p, repeat GSAT scheme, i.e.,
   make the best possible local move.
```

Figure 2.4: GWSAT

The difference between the random noise and random walk strategies is that the former is not restricted to a set of variables that appears in the unsatisfied clauses. An interesting aspect of the SA strategy from [Sel94] is that after flipping it keeps track of changes in the number of unsatisfied clauses for each selected variable. This change is commonly expressed by three variables: $break$, $make$ and $score$. Here, $break$ represents a number of satisfied clauses that becomes unsatisfied after flipping a value of the selected variable x. The variable $make$ counts clauses that are currently unsatisfied, but become satisfied after the flip, and $score$ is computed as $score(x) = make(x) - break(x)$ [Hoo99]. Different definitions of $score$ may be found in the literature. In [Gen93] an increase in satisfied clauses is calculated as the difference between the number of unsatisfied clauses obtained before and the number of unsatisfied clauses obtained after the flip (similar to GSAT [Sel94]). Related to the previously mentioned SA approach with GSAT, [Sel94] has introduced variable δ that, for a randomly picked variable, counts changes in the number of unsatisfied clauses (similar $score$). If $\delta \leq 0$ the variable is flipped. Otherwise, the probability of the flip is determined by $e^{-\delta/T}$, where T is the SA parameter. [Sel95] showed that GWSAT performs significantly better than the random noise and SA strategies.

Various GSAT variants and its improvements have been described in [Gen93]. The HSAT solver incorporates a search history in order to pick the least recently flipped variable. It is shown to work better then the other existing versions of GSAT. Search history within HSAT is commonly depicted as age of the variable x, defined as the number of steps since x was last time flipped.

2.3.4.3 WalkSAT

After the introduction of several SLS solvers in [Sel94] the authors proposed different approach for selection of variables, the WSAT (WalkSAT). The procedure is considered

to be a focused random walk algorithm because the first step after the initial assignment is to choose uniformly at random an unsatisfied clause. From the selected clause a variable is picked either randomly or by greedy rule. Greediness in WalkSAT is identical to GSAT, i.e., variable that leads to the least number of unsatisfied clauses is favored. This greedy property is mostly established by $break(x)$ of a variable x. In Figure 2.5 the pseudo-code of WalkSAT follows guidelines given in [Sel94] (originally, pseudo-code for WalkSAT is not provided). The choice of a variable inside of a randomly picked unsatisfied clause depends on the value of *break* and *noise parameter* p (usually set to 0.5).

```
Pick an unsatisfied clause C.
   For each x ∈ C calculate break(x).
   Calculate u := min_{x∈C} break(x).
   If u = 0
      variable with break(x) = 0 is flipped.
   Else
      With probability p, pick a variable.
      With probability 1 − p, repeat GSAT scheme.
```

Figure 2.5: WalkSAT from [Sel94].

2.3.4.4 Novelty

Most of SLS solvers represent a combination of the focused random walk and the greedy local search. From the introduction of the local search for SAT problems in [Gu92] and (especially) after the introduction of GSAT and WalkSAT in [Sel92], the new state-of-the art algorithms emerged on the grounds of the two architectures [Hoo00a]. The fusion of basic and new ideas gave birth to the *Novelty* algorithm [McA97]. The algorithm incorporates WalkSAT architecture and variable *score* and exploits a search history, i.e., variable *age*. Based on the value of *score* the best and the second best variables are determined. If the best variable is also not the youngest one, its value is changed. Otherwise, with probability p the second best variable is flipped, and with probability $1 - p$ the best variable is selected [McA97].

2.4 Chapter summary

In this chapter we review basic definitions of optimization problems, and, occasionally discuss about the terminology from across the literature. Moreover, we refer to:

- Connection between decision and search problems. We revisit definitions of NP-hard problems based on material in [Gar79].
- Formal definitions of optimization problems, local and global optimum of the objective function, based on material in [Pap98].
- Development of operations research and its intersection with established mathematical fields, such as mathematical programming and calculus of variations.

- Development of various heuristic methods, as integral part of most meta-heuristics.
- Two know combinatorial optimization problems that are also a topic of the dissertation.

CHAPTER 3

Meta-heuristic methods

Both the classical (exact) and heuristic methods have certain limitations. Exact methods usually require a lot of resources, while heuristics may provide sub-optimal solutions of not so good quality. To eliminate these limitations general methods were developed, known as meta-heuristic methods. The generality implies that meta-heuristics do not impose *a priori* knowledge about the problem to be optimized. Therefore, meta-heuristics could be applied to a variety of optimization problems, however, they should be tailored to each particular problem separately. Meta-heuristics iteratively construct and/or improve solutions by applying some predefined stochastic rules. These kind of techniques are designed to escape local optimum, balance exploration and exploitation and/or establish search independent from the initial configuration. In order to implement some of these strategies, one can choose to start with a large number of initial configurations. It is also often possible to use the previously obtained results to improve decisions in the next trail. The implementation might involve moving to worse solutions before reaching an optimum.

The chapter examines formal definitions of meta-heuristics and provides supplementary material for the previous chapter. It covers classification of meta-heuristics in order to help locate the BCO method in a context of other meta-heuristic methods. Definitions of nature-inspired methods and swarm intelligence models are presented. We give a brief overview of the development of artificial intelligence, followed by an overview of the most known representatives of nature- and mathematically inspired meta-heuristic methods.

3.1 Introduction to meta-heuristics

Meta-heuristic should not be considered as an algorithm, but as a set of concepts used to serve as guidelines for tackling an optimization problem [Bir09]. Meta-heuristics were originally developed for solving a class of optimization problems where variables have discrete structure, that is, for problems of combinatorial optimization. The phrase "meta-heuristic" was first coined by Glover in 1986, referring to the *tabu search* [Glo86]. It took at least 10 years before the name was accepted. Before that, meta-heuristics were referred to as *modern heuristics* [Ree93], or *new age algorithms* [Pap98]. In this thesis we have adopted the original notation *meta-heuristic*, although more often in the literature they are specified as *metaheuristics*. In [Glo86] Glover emphasizes the construction of the tabu search as being superimposed on another heuristic, labeling it as "weak inhibition" search with a small number of steps. He referes to the type of im-

posing constraints that produce incomplete search and allow revisits after a short time, unlike branch-and-bound with a "strong inhibition" that impose more rigid constraints. Blum and Roli define meta-heuristics as methods that try to combine basic heuristic methods into higher level frameworks, aimed at efficiently and effectively exploring a search space [Blu03]. They have presented other definitions of meta-heuristics from literature actual up to that time. In this thesis, we endorse one of the definitions proposed by Voß *et al.* [Voß12].

Definition 6 (Meta-heuristic)**.** A *meta-heuristic* is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method. ◇

We also remind on the latest definition provided in [Gas13, pg. 982] by Sörenson and Glover, that defines meta-heuristics as general methods.

Definition 7 (Meta-heuristic)**.** A *meta-heuristic* is a high-level problem-independent algorithmic *framework* that provides a set of guidelines or strategies to develop heuristic optimization algorithms. The term is also used to refer to a problem-specific implementation of a heuristic optimization algorithm according to the guidelines expressed in such a framework. ◇

An important concept of a meta-heuristic method is a trade-off between *exploration* (*diversification*) and *exploitation* (*intensification*) techniques of the search. Exploration refers to the identification of promising areas of a search space, while exploitation is used to intensify the search inside such areas. This concept is a fundamental part of performance of any search algorithm, although meta-heuristics are completely under its influence with the help of its parameterization, which will be demonstrated in Chapter 5. This trade-off balance was noted in many research papers, the oldest dating as early as the 1950s [Box54, Mic04]. George Box in [Box54] presented a simulation of the trade-off process between exploration and exploitation, using role play scenario between "statistician" and "experimenter". Statistician is performing exploitation, while experimenter conducts exploration. In quite an entertaining manner Box presented a general method for maximizing a response function of some k variables (temperature, time, pressure of reaction, etc). He concluded that the methods used for exploitation can only contribute to the current process of exploration of the search space, but must not influence it. In other words, the information of statistician may be considered for the process of exploration but can also be disregarded, which depends on the experience of an experimenter from some previous work.

3.1.1 Classification of meta-heuristics

Meta-heuristics were designed to attack complex optimization problems where classical heuristics and optimization methods have failed to be effective and efficient [Osm96b]. For that reason, numerous meta-heuristic methods have been developed in the past forty years. There are different ways to classify meta-heuristics, i.e., according to the:

1. Origins of the algorithm,

2. Number of solutions used during the search,
3. Randomization,
4. Manipulation over solution.

Other criteria on which basis meta-heuristics can be classified also exist, however, are not covered here. For example, the way meta-heuristics are making use of the objective function (some methods modify objective function during the work). Furthermore, meta-heuristics can be distinguished by their use of search history, whether they use memory or not [Blu03]. For an extensive classification the reader is referred to a material provided by Talbi in [Tal09] and recent survey by [Bou13]. Here, we concentrate on four mentioned classes of meta-heuristics.

- **Origins.** The general behavior of meta-heuristics is based on the principles taken from nature or are based on mathematical principles. Therefore, the most instinctive is to classify meta-heuristics by their origins. Natural processes that have inspired meta-heuristics are: evolution of species, annealing process, insect colony behavior, particle swarms, immune systems and many others. Meta-heuristics with such background are known to belong to the class of *nature-inspired methods*.

  Mathematically founded meta-heuristics are, for example, Tabu Search, TS [Gen10b, Glo97] and Variable Neighborhood Search, VNS [Han10b, Mla97]). These methods include definitions of some metric to measure the distance between solutions, the neighborhoods to distinguish solutions that are close to each other, and local search principles which enable efficient exploration of the solution space.

- **Population.** Another common classification of meta-heuristics, and often taken in the literature as a fundamental, is based on the number of solutions used during the search: one can distinguish methods that are either *single solution*- or *population-based*. Single solution methods are also known as *trajectory-based methods* and typical representatives of this class are: TS, VNS, Iterated Local Search, ILS, [Lou03, Lou10], Simulated Annealing, SA [Kir83, Nik10]. Population-based meta-heuristics incorporate an idea that a certain combination of existing solutions can generate a new solution. Great number of popular nature inspired methods are also population based. Typical examples are Genetic Algorithms, GA [Gol89, Ree10], Bee Colony Optimization, BCO [Luč01, Teo09b], Ant Colony Optimization, ACO [Dor99, Dor10], Particle Swarm Optimization, PSO [Jam95, Pol07], Artificial Immune Systems, AIS [Ber91] and Firefly algorithm, FA, [Yan07b].

- **Randomization.** Classification based on the types of rules used during the decision process, defines two types of meta-heuristics: *deterministic* and *stochastic*. In deterministic meta-heuristics an optimization problem is solved by means of decisions that are based on deterministic set of rules. Such course for solving is typical for TS. In case when decision process is based on randomization, a stochastic algorithm is then generated. This situation is most common as it is related to nearly all of the today's meta-heuristic methods.

- **Handling of solution.** Moreover, meta-heuristics can be classified as *constructive* (if they build new and better solutions during their execution), or based on the

improvement principles (in the case when they transform given solutions in order to obtain the improved ancestors). The representatives of constructive meta-heuristics are Greedy Randomized Adaptive Search Procedure, GRASP [Feo95, Res10], ACO and BCO, while GA, SA, Multistart Local Search (MLS), VNS, PSO as well as some newly developed BCO variants represent methods based on the improvement of given initial solutions. For further details on the design and implementation of meta-heuristic methods the interested reader is referred to [Gen10a, Tal09].

Recently, some researchers have suggested that the increased number of nature-inspired optimization methods has diminished the role of meta-heuristics within the operations research community [Sör13]. Arguments Sörensen has provided was to show the vulnerability behind the misuse of all different possible *metaphors* one could employ in order to develop a novel method. We agree with the author that the time spent to develop new meta-heuristic method might be better used. In particular, a better understanding of mechanisms behind the panel of an existing meta-heuristic might be more valuable. With all that in mind we believe that the research, conducted and presented within this dissertation, contributes to the field of optimization.

3.1.2 Nature- and bio-inspired methods

Many natural phenomena, such as Darwinian evolution, the functioning of the brain, group (swarm) behavior, or the immune system, have fostered new paradigms in computer science. Such interaction between Nature and Computation gave rise to the field of research known as *natural computing* or *nature-inspired computation* [Yao99, Bal99, Cas07]. Natural computing is a study of computational systems that gets ideas and inspiration from natural systems. The most established nature-inspired models of computation are cellular automata, neural computation, and evolutionary computation. Specifically, a class of nature-inspired models that are inspired by biological processes with the purpose to solve hard problems are known in the literature as *bio-inspired computing*.

A rapid growth of natural computing can be explained by a number of factors, such as the boost of a computing power and memory storage capacity, increased number of problems that are influenced by nonlinearity or high-dimensionality [Cas07]. However, the popularity of the bio-inspired algorithms is primarily caused by the ability of biological systems to effectively adapt in frequently changeable environments. Evolutionary computation, neural networks, ant colony optimization, bee colony optimization, particle swarm optimization, artificial immune systems, and bacteria foraging algorithm are among the algorithms and concepts that were motivated by biological processes. For optimization applications in fields such as pattern recognition, self-identity, data analysis and machine learning, nature- and bio-inspired methods are capable of outperforming the classical techniques by providing better and more flexible solution choices [Wan09].

In nature we can find many examples of animal societies, such as insect societies, that exhibit intelligent collective behavior. Still, observing each individual of these societies shows us that they operate without centrally organized control and in unpredictable and dynamic environment [Teo09b]. A global intelligent behavior arises

as a result of the interactions between these individual organisms. Insect society is the classical example of a birth of global effects from local interactions and communication. Analysis of such behavior gave rise to the notion of the *collective intelligence* that is today a basic part of research in different scientific fields, primarily in artificial intelligence, biology and computations.

3.1.3 Swarm intelligence

Swarm Intelligence (SI) is the discipline that deals with natural and artificial systems composed of many individuals [Dor07]. The name for SI was coined by Beni and Weng in the 1989 in order to present Cellular Robotic Systems (CRS) as capable of being intelligent [Ben93]. CRS are collections of autonomous, non-synhronized, non-intelligent robots cooperating to achieve some given tasks [Ben93]. In area of operations research, SI is used to represent models of simple behaviors of the individuals, their interactions with each other and with their environment, which can be applied for coping with difficult optimization problems. Historically, the idea to use the collection of simple agents for solving optimization problems dates back to 1960s [But64]. Back then, the idea of computing with a swarm of simple agents was based on using large numbers of elementary entities (automata) with ability to randomly walk through space, graph or network. It is considered that the Tsetlin's work [Tse73], is the first to address collections of automata as a model for collective behavior of a group with no *a priori* information other than the rules of the game, and attempts to derive the structure of systems that exhibit self-organizing behavior.

SI is considered to be an area of Artificial Intelligence (AI) that is based on studying the actions of individuals in various decentralized systems [Bon99]. However, there is still big discrepancy between two schools of thought in the Artificial Intelligence that are known by the name *neat* and *scruffy* AI. Neat researchers in AI base their work on logic and formal extensions of logic, while scruffy AI researchers use relatively simple approach [Bro11]. SI models are typical for scruffy AI researchers.

When creating SI models and techniques, researchers apply some principles of the natural swarm intelligence. From biology perspective, swarm behavior is based on the biological needs of individuals to stay and work together without any central control. In such a way, it is believed that individuals increase the probability to stay alive, since predators usually attack only isolated individuals. Colonies of various social insects (bees, wasps, ants, termites) are characterized by the swarm behavior. This type of behavior is first and foremost characterized by autonomy, distributed functioning and self-organizing. The most important characteristic here is the self-organizing. It requires interactions (communication) between the individual agents. There are two types of communication among insects: the direct and indirect. Direct communication that is typical for colonies of bees, imply food or liquid exchange and visual contact. The indirect interactions (also known as *stigmergy*) imply modifications of the environment, which in turn modifies behavior of individual agents, such as ants, at a later time. Communication systems between individual agents contribute to the collective intelligence pattern of SI.

In 1990s the group around Mark Millonas from Santa Fe Institute developed mathematical model of the dynamics of swarms and collective intelligence based on pheromone-sniffing, simple-minded ants [Ken01, Lim09]. They presented five basic princi-

ples of SI as:

- The *proximity*: the ability of the population to carry out simple space and time computations;
- The *quality*: the ability of population to respond to environmental factors, such as quality of the food;
- The *diverse response*: the population should be able to distribute resources;
- The *stability*: the ability of population to maintain its mode of behavior against the environment changes.
- The *adaptability*: the ability of population to change the group behavior when it leads to an advancement.

It is a general consensus today that the use of the term “swarm” refers to different systems with a similar architecture. In [Ken01], its description was taken from the Santa Fe Institute document about the swarm simulation system, where it states: "*The classic example of a swarm is a swarm of bees, but the metaphor of a swarm can be extended to other systems with a similar architecture. An ant colony can be considered as a swarm whose individual agents are ants, a flock of birds is a swarm whose agents are birds, traffic is a swarm of cars, a crowd is a swarm of people, an immune system is a swarm of cells and molecules, and an economy is a swarm of economic agents.*"

As of 2004 a new terminology was also presented, namely *computational intelligence*, CI. The CI approaches comprise of all nature-inspired methods. The name was coined by Bezdek [Bez92], however it was firmly established in Institute of Electrical and Electronics Engineers (IEEE)[(1)], by the Computational Intelligence Society (CIS) [Mum09, pg. 5]. The scope of the CIS community, as stated on their web site, is to deal with the theory, design, application and development of biologically and linguistically motivated computational paradigms emphasizing neural networks, connectionist systems, genetic algorithms, evolutionary programming, fuzzy systems, and hybrid intelligent systems in which these paradigms are contained. Extensive informations can be found at `http://cis.ieee.org/scope.html`.

Other names are also used in the research community, such as *soft computing*. Due to lack of standardization in the literature we have opted for the term *swarm intelligence*. All SI algorithms possess several common advantages such as the robustness against the obscure mathematical descriptions and the unique mathematical properties of optimization problems, as well as the capacity to attain global optima [Che13].

3.2 Examples of meta-heuristics

Despite not being used for the experiments in this dissertation, we review several well known meta-heuristic methods. The majority of them belong to the class of nature-inspired algorithms. However, the most popular meta-heuristic methods based on the mathematical principles are also presented, namely VNS and TS. In each class the methods are presented following a chronological order.

[(1)] http://cis.ieee.org/

3.2.1 Simulated annealing

The history of meta-heuristics is connected with a couple of methods already mentioned until now, such as simulated annealing and Evolutionary Algorithms (EA). SA is considered as one of the oldest nature-inspired meta-heuristics, establish on imitation of physical annealing process. It emerged from independent work of two groups of researchers: Kirkpatric, Gelatt, Vecci [Kir83] and Černý [Čer85] who have found the analogy in statistical thermodynamics. When first proposed, it was considered a powerful methodology for solving combinatorial problems, since the search for high quality solutions does not depend on the choice of the initial solution and for some implementations of the algorithm it is possible to derive polynomial upper bound on the computation time [Aar88, Osm96b]. Another important feature of the first SA was the proof of convergence to an optimal solution, however in infinite computational time [Aar88].

Using the concept of annealing as an inspiration dates back to 1953. In particular, it has inspired the construction of the so-called Metropolis algorithm. The algorithm was used for simulating energy changes exhibited during the evolution of a solid to thermal equilibrium. The same scheme inspired SA algorithm, where a particular optimization problem represents a physical system, a solution of the problem is equivalent to states of the physical system, and an objective function of the problem corresponds to energy of the state of the physical system [Aar88, Tal09].

SA is a single solution iterative stochastic algorithm that starts from some initially generated solution, and does not use information gathered during the search [Tal09]. At each iteration two solutions, current ($\boldsymbol{x}$) and newly selected ($\boldsymbol{x'}$), are compared [Nik10]. A solution that is better then the current will always be accepted, while the probability of excepting non-improving solution is a function of SA control parameter, the temperature. The probability to accept a newly selected solution of a minimization problem follows, in general, the Boltzmann distribution, as can be observed in formula (3.1) [Tal09, Nik10].

$$P\{\boldsymbol{x'} \text{ is accepted}\} = \begin{cases} 1 & \text{if } f(\boldsymbol{x'}) \leq f(\boldsymbol{x}) \\ e^{-\frac{f(\boldsymbol{x'})-f(\boldsymbol{x})}{T}} & \text{if } f(\boldsymbol{x'}) > f(\boldsymbol{x}) \end{cases} \tag{3.1}$$

Figure 3.1 presents simplified pseudo-code of SA. At each level of the temperature many trials can be explored until an equilibrium state is reached [Tal09]. At that point the temperature is decreased (according to a cooling rule) in such a way that less non-

```
Initialization phase;
Repeat
    Repeat
        Generate a random neighbor;
        Acceptance phase;
    Until Equilibrium condition;
    Temperature update;
Until (the stopping criterion is satisfied).
```

Figure 3.1: Pseudo-code of the SA algorithm

improving solutions are accepted as the search approaches its end. In addition, the stopping criterion is chosen and can include the maximum total number of iterations, the maximum total number of iterations without improvement of the objective function, the maximum allowed CPU time or any their combination.

3.2.2 Evolutionary computation and genetic algorithm

An important class of meta-heuristic methods are evolutionary algorithms (EA). *Evolutionary computing*, or *evolutionary computation*, EC, is a field of research inspired by evolutionary biology aiming to develop optimization methods for complex optimization problems [Cas07]. The inspiration is drawn from the capabilities of live systems to adapt to challenging conditions in the environment, often described as natural selection or as survival of the fittest. Development of the field of EC is considered as contribution of many researchers starting from late 50s and early 60s, according to [Bac97, Ree10]. Their work introduced different EC paradigms, such as *evolutionary programming* (EP) developed by Lawrence Fogel in 1960, *evolution strategies* (ES) developed by group of three students Bienert, Rechenberg and Schwefel in Berlin during 1960s, and *Genetic Algorithms* (GA) mainly developed by John Holland in the early 1960s [Bac97, Tal09]. This three main forms of evolutionary algorithms were developed independently until 1990s, when EA researchers organized an international workshop devoted to establishing more cooperation in the EC field [Bac97].

Evolutionary algorithms, EA, are population based meta-heuristics that belong to the class of iterative algorithms used for simulation of the evolution of species [Tal09]. Pseudo-code of the EA is provided in Fig. 3.2, however, it should be considered as a broad template covering shared ideas of all the versions of today EAs. Each iteration of an EA corresponds to a *generation*. An algorithm starts with the random creation of the population, where each individual represents an encoded version of some feasible solution. In the evaluation phase an objective function is used to assign to each member of the population a *fitness value*, indicating its suitability to the problem [Tal09]. Following a selection scheme, individuals with better fitness are selected with higher probability for the next operation. The chosen individuals are being *reproduced* by implementing variation operators (e.g., *crossover* and/or *mutation*). In the next stage it is determined which individuals of the population will survive by means of *replacing* the parents by the offsprings in some probabilistic manner (in EP) or in total (in canonical GA).

```
Initialization phase (initial population);
Repeat
    Evaluate population P(t);
    Select fittest individuals for reproduction (P'(t));
    Reproduction of fittest;
    Replacement of parents by offsprings;
Until (the stopping criterion is satisfied).
```

Figure 3.2: Pseudo-code of the EA algorithm.

One of the widely used class of the EAs is the class of genetic algorithms. GAs are

considered as the original prototypes of EAs, which were popularized by Goldberg in late 1980s, as in [Gol89], according to [Blu12]. The current model used for genetic algorithms originated from the studies of adaptive systems conducted by Holland and his colleagues in the 1960s [Hol92]. Holland's GA introduced the idea of recombination (crossover) [Ree10]. GAs are mostly used for discrete optimization problems and are associated with the use of binary representation [Tal09]. However, there are many variations of GA according to the representation of solution, selection strategy, type of crossover and mutation operators, etc., [Bou13].

3.2.3 Ant Colony Optimization

Ant Colony Optimization, ACO, [Dor99, Dor10] is a population-based meta-heuristic inspired by the foraging behavior of ants in nature. The first version of ACO was proposed by Dorigo as an Ant System method (AS) in 1992 for solving TSP [Dor92, Dor10]. As AS didn't manage to be competitive with up to that time specific oriented algorithms designed for dealing with TSP, next years were devoted to development of better version, known today as ACO. This method uses artificial ants to construct solutions by incrementally adding new components [Dor10]. In the analogy to the biological example, the communication between artificial ants is indirect and uses pheromones as a communication medium. The pheromone trails in ACO serve as a distributed, numerical information, which the ants use to probabilistically construct solutions to the problem being solved. Moreover, the ants adapt this information during the algorithms execution to reflect their search experience.

A general outline of the ACO meta-heuristic according to [Dor10] is given in Fig. 3.3. After initializing parameters and pheromone trails, the main loop is performed. It consists of three main steps. In the first step each ant from the colony constructs a solution by selecting its components using a probabilistic rule. This rule takes into account the experience acquired during the search, through the trace of pheromone deposited. The heuristic information of the considered components may also be exploited.

Once the ants have completed their solutions, these may be improved in an optional local search phase. As it was shown in [Dor04, pg. 93], ACO algorithms reach best performance when coupled with local search algorithms. These are used to implement problem specific or centralized actions that cannot be performed by individual ants.

In the third stage, the pheromone trails are updated with an intention to make solution components belonging to high quality solutions more desirable for the following iterations. The pheromone update is performed in two steps. In the first step, the values of the pheromone trails on all the components are decreased by a small factor,

```
Initialization;
Do
    ContructAntSolutions;
    ApplyLocalSearch; //optional
    UpdatePheromones.
While (termination condition not met).
```

Figure 3.3: Pseudo-code of the ACO algorithm.

called the *evaporation rate*. From a practical point of view, pheromone evaporation is needed to avoid premature convergence towards a sub-optimal region. It implements a useful form of forgetting, favoring the exploration of new areas of the search space. In the second step, the values of the pheromone trails on the components belonging to the determined set of high quality solutions are increased depending on the selected evaluation criterion. ACO algorithms usually differ in the way the set of high quality solutions is specified (e.g., current global best solution and/or set of all solutions constructed in the current iteration). When the termination condition is fulfilled, ACO returns the best-so-far (current global best) solution.

3.2.4 Particle swarm optimization

Particle Swarm Optimization (PSO) is a population-based meta-heuristics introduced for a continuous nonlinear optimization problem in 1995 [Jam95]. It emerged from the work of psychologist, J. Kennedy, and electrical engineer, R. Eberhart with an aim to contribute to the field of artificial intelligence [Pol07]. The basic idea is the mimicking of social behavior of bird flocks and fish schools searching for food. This method belongs to the class of SI algorithms as it was inspired by dynamics of population interactions without central coordination.

PSO is an iterative stochastic algorithm that simulates swarm of particles, each representing some candidate solution of the particular optimization problem. It might be viewed as a cellular automata, as the new solution of each particle depends on its old value and the solutions in its neighborhood [Tal09]. Pseudo-code for PSO is provided in Figure 3.4. In the initialization phase, each particle starts from randomly generated solution, after which the iteration begins with the evaluation of the solutions. Since a particle is defined in terms of its position and velocity, an update phase consists of calculating the new position of each particle with an objective to move the search toward the global optimum. This is done by calculating new velocity values as a function of the old one, the position of the particle's best-so-far solution ($\vec{p_i}$), and the position of the best-so-far solution of the entire swarm ($\vec{p_g}$), described by (3.2). After calculating velocity, particle's position is updated [Tal09].

$$v_i(t) = v_i(t-1) + \rho_1 C_1 \times (p_i - x_i(t-1)) + \rho_2 C_2 \times (p_g - x_i(t-1)) \tag{3.2}$$

```
Initialization phase;
Repeat
    Evaluation phase;
    Update phase:
        Update velocities;
        Update solution;
Until (the stopping criterion is satisfied).
```

Figure 3.4: Pseudo-code of the PSO algorithm.

3.2.5 Artificial Bee Colony

The Artificial Bee Colony [Kar05, Kar07] is a population-based meta-heuristic introduced by Karaboga in 2005 as a method inspired by the foraging habits of honeybees. The colony of artificial bees in ABC contains three groups of bees: employed bees associated with specific food sources, onlooker bees watching the dance of employed bees within the hive to choose a food source, and scout bees searching for food sources randomly. Both onlookers and scouts are also called un-employed bees. In ABC, the position of a food source represents a possible solution to the problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. In the basic form of the ABC algorithm, the number of employed bees is equal to the number of solutions since each employed bee is associated with exactly one solution. The general algorithmic structure of the ABC optimization approach is given in Fig. 3.5.

In the initialization phase, the population of solutions is initialized by scout bees and control parameters are set. In the employed bees phase, the neighborhood of each solution is examined in an attempt to find the new solution with a higher quality. In particular, a random solution from the neighborhood is generated, its fitness is calculated and a greedy selection is applied between this and the original solution.

In the next (onlooker bees) phase employed bees share the information about the quality of their solutions with the onlooker bees. Onlooker bees probabilistically choose their solutions, depending on the information provided by the employed bees. For this purpose, a fitness based selection technique can be used, such as the roulette wheel selection method. After a solution for an onlooker bee is chosen, a neighborhood solution is determined randomly, and its fitness value is computed. As in the employed bees phase, a greedy selection is applied between these two solutions.

In the scout bees phase, employed bees whose solutions cannot be improved through a predetermined number of trials (called *limit*) become scouts and their solutions are abandoned. Subsequently, the scouts start to randomly search for new solutions. Hence, those solutions which are initially poor or have been made poor by exploitation are abandoned in order to avoid suboptimal solutions.

```
Initialization phase;
Repeat
    Employed bees phase;
    Onlooker bees phase;
    Scout bees phase;
    Memorize the best solution achieved so far;
Until (the stopping criterion is satisfied).
```

Figure 3.5: Pseudo-code of the ABC algorithm.

3.2.6 Tabu search

Tabu Search (TS) was introduced by Glover in 1986 [Glo86] and it is a single solution meta-heuristic based on the utilization of its search history. However, 10 years earlier, most of the basic elements of TS were suggested by Glover as a part of specific oriented

heuristic for solving nonlinear covering problem [Glo77, Osm96b]. TS is commonly applied to combinatorial optimization problems. It can be viewed as an extension of a classical steepest local search method presented in [Han86], including various types of memory structures (classical being *short-term memory*) in order to overcome local optima [Gen10b]. It shares similarity with the SA algorithm as a method that guides the local search to avoid bad local optima by accepting non-improving solutions. It uses, however, a deterministic rather than stochastic acceptance criterion [Osm96b, Tal09]. The basic functionality behind TS is a dual relationship between imposed *restrictions* and *aspiration criteria* on the TS moves [Glo89]. The restrictions are imposed by constraints which label certain TS moves as forbidden, *tabu*. On the other hand, a search can be released by means of a short-term memory functions. This is achieved by incorporation of an *aspiration level function* $A(s, x)$ used to provide flexibility by overriding the *tabu* status of a move if the aspiration level is attained [Glo89].

The basic version of TS is presented by pseudo-code in Fig. 3.6. The algorithm starts with generation of an initial solution. At each iteration a complete neighborhood of the current solution is searched in a deterministic manner [Tal09]. When a new solution leads to improvement, TS replaces the current solution with the new. If all examined neighboring solutions do not lead to an improvement (TS reached local optima), TS moves to the best admissible neighbor, even if this causes an objective function to deteriorate. This kind of admission criterion may lead to cycling, that is, returning to already visited solutions [Osm96b]. To avoid cycles, TS memorizes attributes[(2)] of previously visited solutions, proclaimed as tabu, by storing them in a so called *tabu list*. The tabu status of a solution is valid for a number of iterations, and this is the reason for referring to tabu list as a *short-term memory*.

```
Initialization phase;
Repeat
    Find best non-tabu neighbor, or satisfy aspiration criteria;
    Update current solution;
    Update tabu list;
    Update aspiration conditions;
Until (the stopping criterion is satisfied).
```

Figure 3.6: Pseudo-code of the TS algorithm.

Sophisticated TS algorithm can hold more information, saved in terms of a *medium-term* and a *long-term* memory. Both types of memories represented advanced mechanisms used to deal with an intensification and diversification of a search. Intensification process concerns storing of best (elite) solutions found during the search in order to guide the search towards promising areas of the search space. Diversification relates to storing of information that diversify the search by discouraging the attributes of some elite solutions. It is important to notice that the diversification of the search is not always useful, as the search is closely connected to a solution landscape of an optimization problem. For example, when all high quality solutions are grouped within a

[(2)] In order to decrease memory used for storing already visited solutions, a subset of information can be used for determining the position/configuration of the solution. The elements of the subset are referred to as *attributes* of the solution.

small distance, diversification can be found useless [Tal09, Gen10b].

3.2.7 Variable neighborhood search

Variable Neighborhood Search (VNS) is a single solution meta-heuristic, proposed by Hansen and Mladenović in 1995 [Mla95, Mla97]. It is a simple and an effective optimization method which has been widely used for dealing with combinatorial and global optimization problems [Han10b]. The VNS search is based on a systematic changing of neighborhoods within a descent phase in order to find a local optimum, as well as within a perturbation phase to get out of the corresponding valley [Han10b, Cra14]. It uses multiple neighborhoods in order to increase an efficiency of a search, and is based on three simple properties [Han03]:

1. A local optimum w.r.t. one neighborhood structure is not necessarily a local minimum for another;
2. A global optimum is a local optimum w.r.t. all possible neighborhood structures;
3. For many problems, local optimum w.r.t. one or several neighborhoods are relatively close to each other.

A basic building block of VNS is LS procedure. In order to describe VNS, an introduction into particular notation is first required. For an arbitrary solution $\boldsymbol{x} \in X$ a neighborhood of $\boldsymbol{x}$ ($\mathcal{N}(\boldsymbol{x})$) is a set of all solutions obtained from $\boldsymbol{x}$ by an application of some predefined elementary transformation. Let $\mathcal{N}_k$, $(k=1,\ldots,k_{max})$ be a finite set of pre-selected neighborhood structures. Then, $\mathcal{N}_k(\boldsymbol{x})$ is a set of solutions in the k^{th} neighborhood of $\boldsymbol{x}$, i.e., the set of solutions obtained from $\boldsymbol{x}$ by the application of k elementary transformations [Cra14]. A general outline of the VNS is presented in the form of pseudo-code in Fig. 3.7.

In the initialization phase an initial solution is usually determined by some constructive heuristic and improved by LS before the beginning of the actual VNS procedure. In a *shake* procedure a random point $\boldsymbol{x'}$ is generated in a k^{th} neighborhood of $\boldsymbol{x}$, $(\boldsymbol{x'} \in N_k(\boldsymbol{x}))$. The role of the *shake* procedure is to prevent trapping in a local optimum. Intensification of a search is realized by the *improvement* step which involves the selected LS procedure with an aim to improve a current solution $\boldsymbol{x'}$ in order to obtain a local optimum $\boldsymbol{x''}$. In the *move* step, if this local optimum is better than the current best-so-far solution, the search moves there $(\boldsymbol{x} = \boldsymbol{x''})$, and continues within $\mathcal{N}_1$ $(k = 1)$. More precisely, the entire VNS procedure is concentrated around the current global best solution, and therefore, *move* step has to ensure that this solution is always updated, as early as possible. If local optimum is not better than the current global best solution, algorithm moves to the next neighborhood $(k = k + 1)$. Once the stopping criterion is met, the global best solution is reported [Tal09, Han10b, Cra14].

Basic VNS has a unique parameter k_{max} – the maximum number of neighborhoods. Sometimes, but not necessarily, successive neighborhoods are nested. There are several variations and modifications of this basic VNS scheme, as well as many successful applications. The readers are referred to [Han10b, Han14] for more details.

Initialization phase;
Repeat
  Set $k = 1$.
  Repeat
    Shake;
    Improve;
    Move;
  Until $k > k_{max}$;
Until (the stopping criterion is satisfied).

Figure 3.7: Pseudo-code of the VNS algorithm.

3.2.8 Final remarks

A detailed classification of various types of meta-heuristic is given in a recent survey by Boussaïd *et al.* [Bou13]. Beside development of mathematically-based meta-heuristics, during the last decades behavior of social insects became an inspiration for design of new SI artificial systems. Here, we name a few algorithms that have appeared in the last 15 years and are based on a behavior of bees: Bee System [Luč01, Luč03b], Bee Colony Optimization (BCO) [Teo05], Marriage in Honey-Bees Optimization (MBO) [Abb01], BeeHive [Wed04], Honey Bees [Nak03], Artificial Bee Colony (ABC) [Kar05], Bee System Optimization (BSO) [Dri05], the Bees Algorithm [Pha06], Honey Bee Marriage Optimization (HBMO) [Afs07], Fast Marriage in Honey Bees Optimization (MHBO) [Yan07a], Virtual Bee Algorithm (VBA) [Yan05]. In all these variants numerous agents (i.e., artificial bees) investigate search space at the same time. However, a central concept behind their work is the cooperation, allowing them to achieve better results from those that would otherwise be generated by the independent work. In this dissertation, special consideration is given to the Bee Colony Optimization method. The primary goal is to provide detailed description behind its development and underlying mechanisms by employing tools of theoretical and empirical analysis.

3.3 Chapter summary

In this chapter we review different schools of meta-heuristic design. Meta-heuristics have proven their power in obtaining high quality solutions to many complex real world problems. The structure of this chapter can be reviewed by following the list of presented topics:

- Definition of meta-heuristics and elaboration on different types of meta-heuristic classifications according to the: origins of the algorithm, number of solutions used during the search, randomization and manipulation of solutions.
- A brief interpretation of the bio-inspired methods and swarm intelligence.
- Overview of the couple of most representative nature-inspired methods, as well as two methods based on mathematical principles.

Topics that were not covered are, e.g., hybrid meta-heuristics that try to combine best features of different methods in order to derive more powerful new algorithms. There are two main classes of hybrid methods: the first one obtained when combining two or more meta-heuristics (e.g., GA with SA or TS, SA with TS), while the second class covers combinations of meta-heuristics with exact methods (e.g., TS with B&B, VNS with MIP) [Blu08, Man09]. It is an ongoing topic that is attracting the attention of an increasing number of researchers, however, it is beyond the scope of this dissertation.

CHAPTER 4

Bee colony optimization method

This chapter is devoted to the Bee Colony Optimization (BCO) method. We start with a short historical exposé about the discovery of certain behavioral pattern of bee swarms in the early 1940s. The goal is to remind of first steps within the development of BCO, inspired by previous research on foraging of bees. In section 4.1.2 we describe mathematical model of BCO. Then, we review two BCO variants and describe steps that led to the current versions of the BCO method. Both variants of the BCO algorithm are reviewed in great detail in section 4.2. Part of our contributions in this thesis is the introduction of *loyalty functions* in section 4.2.3.1, as a characterization of the probability decisions inside of BCO. Finally, we list various applications of the BCO method and provide a short overview of future research topics.

4.1 The development of BCO

Bee colony optimization is a population-based meta-heuristic method that was first proposed by Lučić and Teodorović in 2001 [Luč01]. In the early stages of its development BCO showed promising results in application to combinatorial optimization problems, drawing attention of many researchers around the world [Luč01, Luč02b, Luč03a, Teo05]. BCO belongs to the class of SI algorithms and is first of its kind where basic principles of collective bee intelligence were used in dealing with combinatorial and continuous optimization problems. The inspiration for creating new multi-agent system, such as BCO, originates from the foraging behavior of honey bees. This behavior is suitable for modeling as the practice of collecting and processing nectar is highly organized [Teo09b]. The first version of the BCO algorithm was developed as a constructive procedure, where each artificial bee is building a solution from scratch. Later variant of BCO, known as improvement BCO, used a modification of complete solutions [Dav15b]. To provide better understanding of the BCO structure, the introduction into the behavior of bees in nature is being presented.

4.1.1 Bees in nature

In nature, honey bees succeed to find nectar among limited resources in quite efficient manner, without control of any central management and within unpredictable and dynamic environment. The reason for such success is the capacity for communication using skills that most resemble to symbolic language, a property known only for bees to share [Gou75]. Observations of the bees behavior dates back to ancient times. The first

documented observations about the bees behavior can be found in the Aristotle book "History of Animals" from 350 b.c. However, it took a long time until true research was conducted. Before 1940, researchers recognized that some of the bees that have returned to the hive perform a certain dancing ritual [Hai10]. However, it was Karl von Frisch that in the mid-1940s first realized functionality of what was considered almost unimportant - the *waggle dance* (Fig. 4.1) [VF74]. He earned the Nobel Prize in 1973 for this discovery. As it seems, he has inspired a great number of researches who thereafter decided to deepen understanding of the bees behavior [See85, Cam91]. von Frisch discovered that bees notify their fellow bees about the food sources, i.e., that they communicate. In the survey of von Frisch's life [Hai10] Tania Munz recognized that such discoveries, together with discoveries of other animal communication, have influenced self-image of humans. Until then, it was thought that *homo sapiens* was clearly characterized by the skills of communication, seen as a boundary that divides us from other living organisms on Earth.

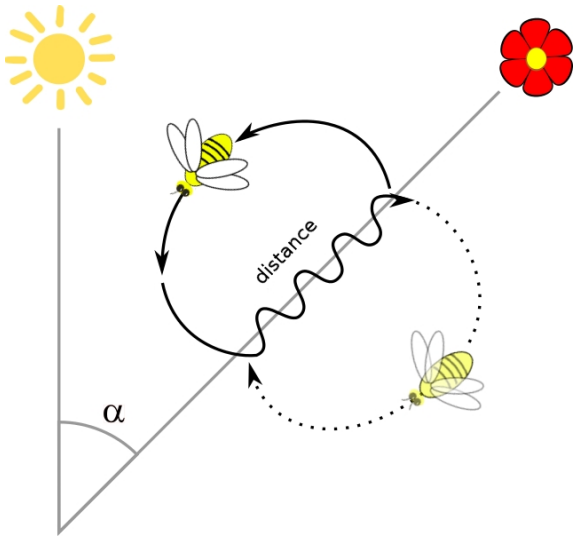


Figure 4.1: Waggle dance of forager bees in nature.

The waggle dance (also known as wag-tail or figure-eight dance), together with the odor, defines a dancing language. Several other forms of the dance were recognized, such as a *round dance* [Sam98]. Round dance is the simplest and doesn't provide much information, except that the food is somewhere close to the hive. The bees use waggle dance to learn about different properties of a food source, such as a position (defined by the direction and distance) [Com11]. The direction is specified with the angle α between food source and the Sun (Fig.4.1). The distance is represented by the bees' shaking of hindquarters ("waggle"). The odor is used to help other foragers to determine the quality of the food source. In the rest of this thesis both parts of the dancing language will be referenced as only a *dance* or a *waggle dance*.

Understanding the mechanism behind the foraging of honey bees colonies was already proposed in the middle of 80s, from which here we distinguish work of Seeley [See85]. Seeley was inspired by progress made in models of bumble bees. He recognized that even substantial work has been done to understand the inner working of

honey bees colonies, a lot still remained to be discovered [See95]. The initial study, he conducted together with his colleague Kirk Visscher, reviled a great deal of new insights, like those that bees make individual assessments of a food source's absolute quality, or that honey bees can cover more than 100 square kilometers around the hive during foraging. In joint work with Camazine and Sneyd, Seeley showed that waggle dance of worker bees is changing with regard to an absolute quality of a food source [See91]. When the quality of nectar is high, the foraging bees dance longer and more vigorously [See00]. It can be noted that to conclude a mathematical model of foraging mechanism is quite an ambitious task. Main problem is that much of the nectar collection process of honey bees may not be correctly described [See95]. However, a pursuit towards simplification of the living processes can also lead to interesting discoveries and tools that can be used for solving hard real-world problems.

The first mathematical model of how information of food sources are stored within a honey bee colony was introduced by Camazine and Sneyd [Cam91]. They postulated that the collective memory is shared via the dance language and presented a model that describes the colony's decision-making process using system of non-linear differential equations that have most effect on the total foraging success. This model, however, disregarded some aspects of honey bees foraging process, like individual behavior. In the literature this problem has been addressed, leaving an open question about how important is the *dance language* for communicating memory. In [Gra12] experimental results have shown that the drop in total foraging success, when the use of dance language was removed, was not as drastic as proposed in models of Seeley. Furthermore, another interesting question is the imprecision of information being shared, which can influence on fertility of new food sources of high quality.

The complexity of an insect behavior still today cannot be replicated with mathematical models. Nonetheless, the imitation of only a small part of some observed behavior in animals can, with its natural ingenuity, contribute greatly to the science. The particularity of such knowledge is that otherwise it could not be invented by classical approaches of mathematics or physics.

4.1.2 BCO model

Basically, a mathematical model of the foraging behavior of honey bees can be described as follows [Dav15b]. Bees that are searching and collecting the nectar are known as *worker bees*. They collect and accumulate food for later use by other bees. The worker bees that are exploring the area are called *scout bees*. Typically, scout bees in nature look for food by exploring the fields in the neighborhood of their hive. After completing the exploration, scout bees return to the hive and inform their hive-mates about the locations, quantity and quality of the available food sources in the areas they have examined. In the case they have discovered nectar in the previously investigated locations, scout bees dance in the so-called *dance floor area* of the hive using a ritual called *waggle dance*, in an attempt to attract the remaining members of the colony to follow their lead. If a bee decides to leave the hive and collect the nectar, it follows one of the dancing scout bees to the previously discovered patch of flowers. After taking a load of nectar, the foraging bee returns to the hive, leaving the nectar at a food store. The foraging bee then decides for one of the several scenarios: (1) it can try to recruit its hive-mates with the dance ritual before returning to the food location; (2) it can

continue with the foraging behavior at the discovered nectar source, without recruiting the rest of the colony; (3) it can abandon the food source and become *uncommitted* follower [Cam91, Bon99]. As several bees may be attempting to recruit their hivemates on the dance floor area at the same time, it is unclear how an uncommitted bee decides which recruiter to follow. The only obvious fact is, quote: *the loyalty and recruitment among bees are always a function of the quantity and quality of the food source* [Teo09b, Dav15b].

The described model has served as a basis for BCO algorithm. Here, the artificial bees are considered to be homogeneous, that is, all bees can be considered as being analogous to foragers that can do the work of a scout bee. A general structure of the BCO algorithm, that combines both constructive and improvement version, is as follows. The population of *artificial bees* (B individuals) searches for an optimal solution. Every artificial bee generates one solution to the problem. Each algorithm step consists of two alternating phases: *forward pass* and *backward pass*. During each forward pass, every bee is exploring a search space. It applies a predefined number of moves to either construct the part of a solution [Teo05] or modify an existing complete solution [Dav11b]. During backward pass, bees exchange information about the found solutions after which each bee can decide for one of the two scenarios, mentioned earlier (first or third).

4.1.3 The evolution of BCO

It all started as a *Bee System*, by a joint work of Lučić and Teodorović in 2001 [Luč01]. The development of this algorithm was also a building block for future development of smart algorithms that use basic principles of collective bee intelligence for solving optimization problems [Teo05]. A Bee System, presented in 1997 by Sato and Hagiwara in [Sat97], was introduced as a modified genetic algorithm where communication and dancing was utilized to replace steps of GA, such as mutation and crossover operators. Namely, each bee corresponds to a chromosome of GA that tries to find a high quality solution independently from others. When best solution is found, other bees try to improve it in the local search step of the algorithm. The *Bee System* of Lučić and Teodorović was constructed having in mind only the behavior of bees in nature and incorporating principles of foraging [Luč01, Luč02b, Luč03a]. It was first used on instances of the TSP, where, process of searching for shortest tour greatly depended on the location of the hive (starting position of artificial bees) and the probability model by which bees are choosing the next node. The starting position was randomly selected. The probability model incorporated: distance between the current node (hive at the beginning) and node-candidate to be visited; the total number of performed iterations in a search process; and the total number of bees that have visited the considered link in the past [Teo15]. Results in paper [Luč01] showed that the *Bee System* method can produce optimal solutions in reasonable time. Besides transportation problems [Luč02b, Luč03a], *Bee System* was also used on a routing problems. More precisely, Vehicle Routing Problem (VRP) with stochastic demands [Luč02a]. VRP is a problem of finding a set of routes that would minimize transport costs. The uncertainty in demands arises when the real value of demand becomes known only after the vehicle has reached the node. In a thesis of Lučić [Luč02a], a goal was to generate an intelligent vehicle routing system capable of providing decisions that would lead to a set of high

quality routes. The approach to solve stochastic VRP problem was dealt by using two steps in order to combine advantages of solving TSP with newly developed *Bee System* and fuzzy rules. In the first step, *Bee System* was used for solving the VRP problem, as a TSP, in order to create a *Giant route*. The second step consists of decisions when to finish one's vehicle route and when to start with the next, while walking along the created giant route. The task of deciding whether the vehicle should serve the next node or not was conducted using fuzzy rules. In a similar fashion, on the same problem Lučić and Teodorović in [Luč03b] presented results for combination of *Bee System* and fuzzy logic where decisions were made in real-time with regard to the route shape.

Version of BCO, commonly used today, was proposed in 2005 by Teodorović and Dell'Orco in [Teo05], as an improved and more general version of the *Bee System*. To illustrate the performance of the newly proposed BCO, authors used Ride-matching problem. Ride-matching is a problem of finding the best combination between the vehicle and the passengers who will share the ride, taking into account: vehicle capacity, days in a week when person is ready to have a ride, trip origin, trip destination, desired departure and/or arrival time. In their article, Teodorivić and Dell'Orco assumed that the arrival and departure times are fixed. On a relatively small example of 97 passengers, and a constraint of 4 persons per vehicle, 6 feasible solutions were presented. These preliminary results showed that the development of bee inspired methods could significantly contribute to the solution of complex problems [Teo05].

Bee System had more similarities with the behavior of bees in nature, compared to the BCO method. The main difference between them is that the location of a hive had a bigger role in the *Bee System* model. During the search process, the location of a hive could be changed. Another big difference was that *Bee System* incorporated Logit-based model for the probability of choosing next node, while BCO method is using roulette wheel [Dav15b, Š11].

4.2 The BCO Algorithm

The BCO method is based on engaging a group of *artificial bees* (B individuals) in search for an optimal solution of a considered optimization problem. All artificial bees are engaged in a search process in such a way that each bees generates one solution for a considered problem. For the sake of further analogy, we introduce a *foraging cycle*. One foraging cycle corresponds to a work done by foragers between two adjacent visits to the hive. Namely, after leaving the hive forager bees search for a food source after which they return to the hive to advertise found locations. The process of repetition of the foraging cycle in nature is terminated when the night falls. Similarly, the search process of artificial bees is conducted through iterations, during which bees also communicate in order to compare the quality of obtained solutions, until some predefined stopping criterion is satisfied. In regard to this clear distribution of tasks of artificial bees, each iteration of a BCO algorithm can be represented as a composition of alternating phases (steps): a *forward pass* and a *backward pass*. Therefore, one alternation corresponds to one foraging cycle.

During the forward pass, all artificial bees explore a search space. The method of exploration highly depends on concrete implementation of the BCO algorithm. Namely, the exploration is performed through a certain (predefined) number of moves to ei-

ther construct a part of a solution [Teo05] or modify an existing complete solution [Dav11b]. These moves may be implemented in various ways, for example, by employing some well-known problem-specific heuristic procedure. Furthermore, in the forward pass all artificial bees perform an exploration independently from each other, and therefore, no information is being exchanged in this phase. A number of moves within one forward pass can be represented as a function of a second BCO parameter, namely, NC. Parameter NC is used to determine a frequency of information exchange between artificial bees, therefore, influencing an exploitation of the search. In early (constructive) implementations, NC was employed to count the number of components to be added to partial solution within a forward pass [Dav15b, pg. 41]. To enable a unique description for both BCO variants, in recent implementations of the constructive BCO the definition of NC has changed to represent a number of forward/backward alternations during an iteration. At the end of a forward pass, a new (partial or complete) solution is generated for each bee [Dav15b].

During the backward pass of the BCO algorithm all artificial bees share the information about quality of discovered solutions. In nature, sharing of information is made by performing a dancing ritual where bees notify others about the amount and quality of food, and its proximity to the hive. The information being exchanged in the BCO algorithm contains a quality of each (partial) solution, with respect to the best and the worst solution. After the evaluation of all (partial) solutions, each artificial bee decides, with a probability depending on the solution quality, whether it stays *loyal* to its solution or not. The bees that remain loyal to its solution become *recruiters*, while the rest become *uncommitted*. Consequently, within each backward pass all bees are divided into two groups: R recruiters, and the remaining $B - R$ uncommitted bees [Dav11b]. Values for R and $B - R$ fluctuate from one backward pass to another. In Fig. 4.2 recruiters are marked with a style of von Frisch experiments, who has distinguished recruiters from other foragers with different colors. Next, the uncommitted bees have to select among solutions advertised by the recruiters. A selection process for one of the advertised solutions is stochastic, in such a way that better solutions are given higher probabilities to be chosen for further exploitation (Fig. 4.3).

An illustration of the recruiting process for $B = 10$ is demonstrated in Fig. 4.3. The recruiters are marked with a style of von Frisch experiments, who has distinguished recruiters from other foragers with different colors. After evaluating (partial) solutions,

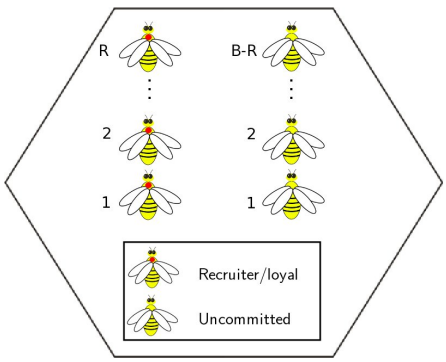


Figure 4.2: Recruiters and uncommitted bees.

bee B_1 decides to abandon its own and join bee B_2. This scenario corresponds to one in nature when second bee has performed waggle dance, in order to advertise a superior food location, after which both bees fly out to discovered location. Within BCO, artificial bee B_1 associates (copies) a (partial) solution of the bee B_2. We demonstrate in Fig. 4.3 possible scenarios for other artificial bees. For example, bees B_6, B_7 and B_8 decided to copy a (partial) solution of bee B_5. Therefore, the quality of a (partial) solution of the bee B_5 was probably much better then of other recruiters.

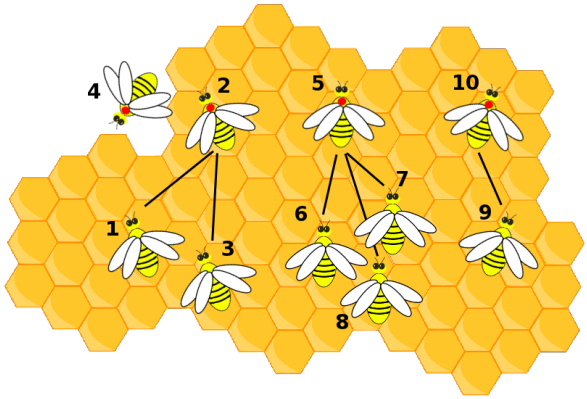


Figure 4.3: Illustration of the process of recruitment.

In nature, bees start their exploration by leaving the hive. However, the hive as an artificial object is not commonly used in BCO and does not influence algorithm execution. It is used only to mark synchronization points at which bees are exchanging information about the current state of the search. To summarize, two phases of the BCO algorithm, the forward and backward pass, alternate in order to generate new and better solutions (one for each bee). Among all complete solutions the best is identified and used to update a *global best solution*. The BCO algorithm runs iteration by iteration until a stopping criterion is met. A possible stopping criterion can imply, for example, a maximum total number of iterations, a maximum number of iterations without improvement of a current global best solution, a maximum allowed CPU time, maximal number of objective function evaluations, etc. [Teo09a, Dav15b].

4.2.1 Pseudo-code for BCO

One among various advantages of the BCO algorithm is the small number of parameters:

- **B** - number of artificial bees involved in search;
- **NC** - number of forward/backward passes during one iteration.

A pseudo-code of the BCO algorithm is provided in Figure 4.4. Steps (a) i (b) in forward pass and step (ii) in backward pass are problem dependent and should be resolved in each implementation of the BCO algorithm. On the contrary, other steps of BCO are problem independent. These steps specify loyalty decision (step (iii)), recruiting process (step (iv)), and update of the global best solution (step (v)), and are therefore described in more detail in the following.

Initialization: Read problem data, parameter values (B and NC), and stopping criterion.
Do
(1) Initialize a solution for each bee.
(2) *For* $(u = 0; u < NC; u++)$
//**forward pass**
(i) *For* $(b = 0; b < B; b++)$
(a) Evaluate possible moves;
(b) Choose a move using a heuristic rule;
//**backward pass**
(ii) *For* $(b = 0; b < B; b++)$
Evaluate the (partial/complete) solution of bee b;
(iii) *For* $(b = 0; b < B; b++)$
Loyalty decision for bee b;
(iv) *For* $(b = 0; b < B; b++)$
If (b is uncommitted), choose a recruiter by a selection rule .
(v) *If* (solutions are completed)
Use the best one to update x_{best} and $f(x_{best})$.
While stopping criterion is not satisfied.
return $(x_{best}, f(x_{best}))$

Figure 4.4: Pseudo-code for BCO

4.2.2 Variants of the BCO algorithm

The BCO algorithm was evolving during the last 15 years. Throughout the evolution two of its variants have developed, *constructive* BCO (BCOc) and *improvement-based* BCO (BCOi) [Dav15b]. The first approach is based on *constructive* steps in which bees build solutions piece by piece. BCOi, more often used today, is based on *transformations* of complete solutions in order to obtain the best possible final solution. The main differences between these two are within forward pass implementations, whereas the backward pass of both algorithms follows the same set of steps. In the text to follow we give a general description for both variants of the BCO algorithm.

4.2.2.1 Constructive BCO

The BCOc method was developed first. It was originally proposed by Lučić and Teodorović [Luč01, Luč02b, Luč03a] and later on used in [Dav09, Luč03b, Mar07, Šel10, Won10a, Won09]. In the BCOc algorithm a construction procedure starts from an empty solution. A number of forward/backward passes depends on a number of components constituting a complete solution. Moreover, the number of constructive moves in a single forward pass is mostly determined as a function $f(NC)$, being restricted by parameter NC. For example, if a complete solution consists of n components, approximately n/NC of them are added to a current partial solution during each forward pass. An iteration of BCOc is finalized once each bee generates a complete solution to a given problem. Then, the best solution is determined and is used to update a global best solution. At this point, all B solutions are deleted and an initialization of a new iteration is performed.

The construction approach usually implies a constructive heuristic procedure that is building a solution step by step, applying some stochastic, problem specific rules, thus generating a (partial) solution. Suppose we have B bees (Bee 1, Bee 2, ..., Bee B) which participate in a decision-making process on n entities. A possible situation that may arise after a first forward pass, where three components are added, is illustrated in Fig. 4.5.

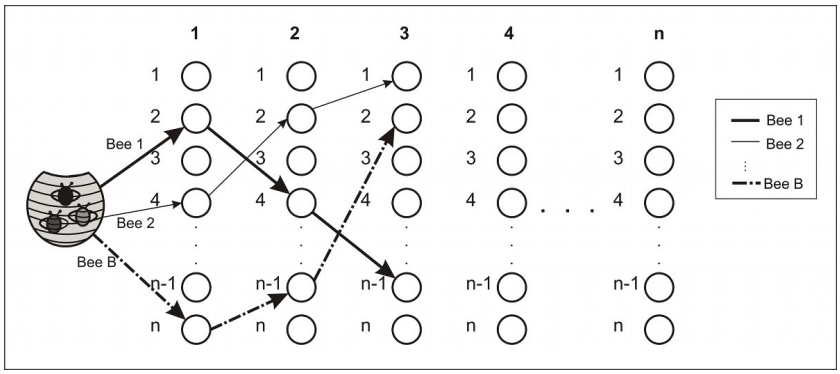


Figure 4.5: An example of partial solutions after the first forward pass.

Upon constructing new partial solutions for each bee, the algorithm starts the backward pass. Based on the probability that depends on solution quality, every bee makes the decision whether it will stay loyal to its solution or not. Let us assume that after comparing all generated partial solutions, Bee 2 from example in Fig. 4.5 decided to abandon its solution and join Bee B. In practice, this implies that the partial solution generated by Bee B is copied to Bee 2. Let us also assume that the Bee 1 decided to stay loyal to its partial solution and, without being chosen by any forager, it performs a new constructive step independently. A possible situation resulting after the first move (decision process) of the second forward pass is illustrated in Fig. 4.6.

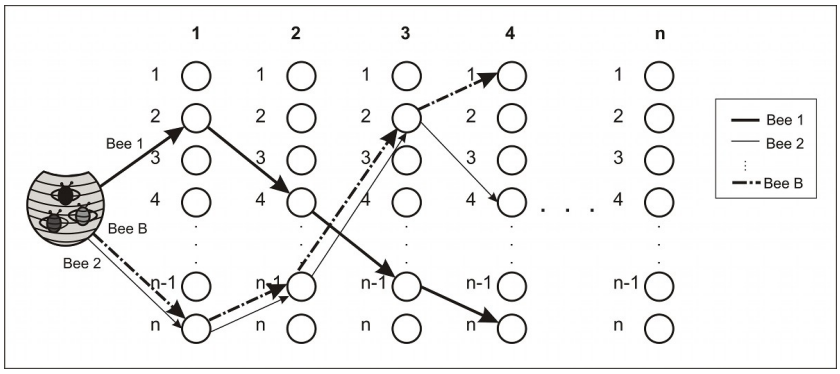


Figure 4.6: An example of partial solutions after the first move in the second forward pass.

Different values of parameter NC can greatly influence the quality of a reported solution. If NC is large, the search process is intensified and each bee generates similar results. On the other hand, if NC is small, each bee adds more components to its partial

solution, thus introducing variety among different solutions.

4.2.2.2 Improvement BCO

BCOi was first proposed by Davidović *et al.* in 2011 [Dav11a]. Some indications of combining constructive with improvement approach were suggested by [Teo09b], however, were not published until 2013 in [Tod13]. BCOi was developed with an aim to overcome unsatisfactory results obtained for a few combinatorial problems dealt with BCOc [Dav15b]. Unlike BCOc, the BCOi algorithm always works on the complete solutions. At the initialization stage, one complete solution (for example, generated randomly) is assigned to each bee. During each forward pass all bees modify several components of their complete solutions in order to enhance them. The role of parameter NC in BCOi is to define maximum number of forward/backward passes in the single iteration. The number of transformations in a single forward pass is usually determined randomly [Dav11a], or as a function of problem size [Sto15]. In particular, it is up to an implementation to ensure that distinct bees perform different modifications. This property is important in order to have different treatment of the same solutions assigned to a recruiter and its follower(s) after the backward pass. The possible situation after s-th forward pass is illustrated in Fig. 4.7.

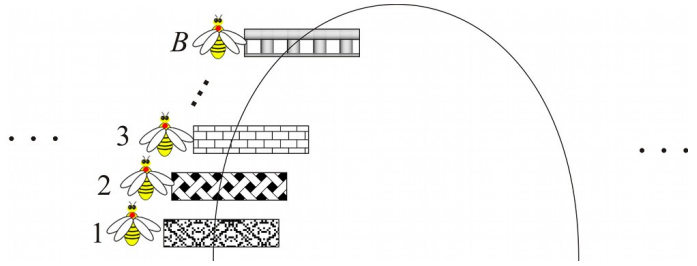


Figure 4.7: An illustration of the s-th forward pass

Rectangles in Fig. 4.7 represent complete solutions associated with bees. Distinct patterns are to denote that each bee is associated with a different solution. Suppose that after comparing all generated (complete) solutions, Bee 1, from Fig. 4.7 decide to abandon its own solution and joins Bee B. Similarly, Bee 4 was recruited by Bee 2 (see Fig. 4.8). Bee 3 decides to stay loyal to its solution.

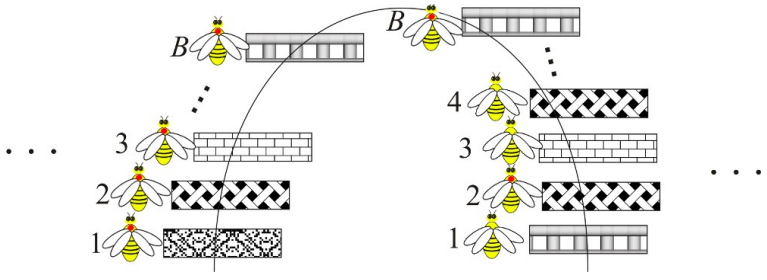


Figure 4.8: The possible result of a recruiting process within s-th backward pass

Next, they are free to make individual decisions about the next step to be made, i.e., to

perform transformations that change each solution in a different way. This is illustrated in Fig. 4.9.

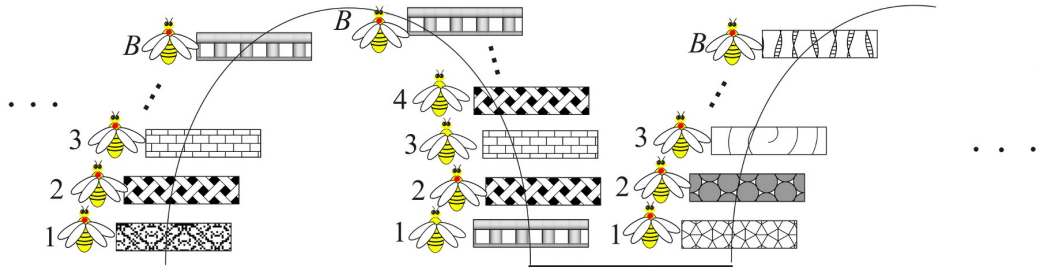


Figure 4.9: An example of complete solutions after $(s+1)$-th forward pass

Considering that the transformations are to be executed for each of the B bees during one forward pass, the reasonable requirement is that the modification must be efficient. Obviously, local search is not an option. A less computationally extensive stochastic transformation may be employed, such as: select a component randomly, change its value or replace it with another component.

Contrary to the BCOc, in BCOi the best solution is used to update a global best solution after each transformation. An iteration of BCOi completes after each bee performs NC solution modifications ($NC \cdot f(NC)$ transformations). The initial solutions for new iteration can be generated in some random manner [Dav11a, Nik13b] or an existing solutions is used [Nik13a, Sto15].

4.2.3 Backward pass and loyalty functions

As mentioned in previous paragraphs, backward pass is a phase where artificial bees share information about the quality of discovered (partial) solutions by following three steps: 1. Evaluation; 2. Decision on loyalty; 3. Recruitment. Before bees can share information the quality of (partial) solutions, obtained in the forward pass, needs to be evaluated. Next, the obtained values are utilized within the decision-making process on loyalty. Once determined, loyal bees become recruiters which in the last step of the backward pass try to recruit uncommitted bees.

Evaluation. If $C_b\,(b = 1, 2, ..., B)$ denotes a quality (value of an evaluation function) of the b-th bee (partial) solution then, in a case of minimization, a normalized value of the C_b is calculated as follows:

$$O_b = \begin{cases} \frac{C_{max} - C_b}{C_{max} - C_{min}} & \text{if } C_{max} \neq C_{min} \\ 1 & \text{if } C_{max} = C_{min} \end{cases}, \quad b = 1, 2, ..., B, \tag{4.1}$$

where C_{min} and C_{max} represent minimal (the best) and maximal (the worst among all bees) value of the evaluation (objective) function among all engaged bees. The normalization in the case of maximization, whereby minimal value (C_{min}) corresponds to a worst and maximal (C_{max}) to the best value, is calculated as follows:

$$O_b = \begin{cases} \frac{C_b - C_{min}}{C_{max} - C_{min}} & \text{if } C_{max} \neq C_{min} \\ 1 & \text{if } C_{max} = C_{min} \end{cases}, \quad b = 1, 2, ..., B. \tag{4.2}$$

The normalized value O_b reflects the quality of a reported solution with regard to objective of the optimization problem. Moreover, it is essential for the next phase of backward pass, i.e., decision-making on loyalty.

Loyalty. In most of BCO implementations, a probability that b-th bee (at the beginning of the new forward pass) will stay loyal to its previously generated partial solution is expressed as:

$$p_b^{u+1} = e^{-\frac{O_{\max} - O_b}{u}}, \quad b = 1, 2, \ldots, B, \tag{4.3}$$

where $O_{\max}$ represents maximum over all normalized values of (partial) solutions to be compared, and u is a forward pass counter (e.g., $u = 1$ for the first forward pass, $u = 2$ for the second forward pass, etc.). To remind, parameter NC in the BCO algorithm implies the number of times the probability function should be invoked. Furthermore, normalized value $O_{\max}$ is, generally, equal to 1. Index $u + 1$ in expression p_b^{u+1} is typically used term in the literature on BCO algorithms. However, after careful consideration the index notation should instead hold u in order to keep consistency in both the format and the description of the algorithm steps. Consequently, formula (4.3) can be rewritten as follows:

$$p_b^u = e^{-\frac{1 - O_b}{u}}, \quad b = 1, 2, \ldots, B. \tag{4.4}$$

For a better understanding of the connection between the evaluation and the loyalty decision process, we offer couple of remarks. Let us consider a problem of minimization of an objective function. When a quality of b-th bee (partial) solution is equal to maximal value C_{max}, its normalized value O_b (Formula (4.1)) takes value 0. However, when the quality is close to C_{min} the normalized value O_b approaches 1. Moreover, if the bee b found the best (partial) solution, its $O_b = 1$. Utilizing equation (4.4) for each artificial bee it is decided whether to become uncommitted follower or to continue exploring already known solution. In its form equation (4.3) assures that the bee b will stay loyal with a higher probability to discovered (partial) solutions of a good quality (the ones with higher O_b value). It is thus obvious that a larger value of O_b (closer to 1) corresponds to a (partial) solution of good quality, inducing higher probability of a bee to remain loyal. As the search process advances (u increases) the influence of already discovered (partial) solution increases, i.e., the probability that the bee will keep and advertise its current solution takes larger values. The influence is expressed by counter u in the denominator of the exponent of expression (4.3). Increasing index u is increasing the influence of already discovered (partial) solution at the beginning of the search. This, again, agrees with the behavior of bees in nature where valuable food locations are well exploited. In addition, increased effort that a bee invests in a current (partial) solution decreases its willingness to abandon it.

Formula by which probability p_b^u is calculated is similar to the one for SA algorithm. In SA it is used to decrease the probability of selecting a solution that is worse. Within the BCO algorithm, the case is to some extant reversed: as the solution is being constructed (u increases), the probability to accept better result is decreasing as bee is more likely to stay loyal to its solution.

Recruitment. The probability that any uncommitted bee chooses (partial) solution of a recruiter r equals to:

$$p_r = \frac{O_b}{\sum_{k=1}^{R} O_k}, \quad r = 1, 2, \ldots, R, \tag{4.5}$$

where O_k represents normalized value of the k-th advertised (partial) solution quality and R denotes the current number of recruiters. Using equation (4.5) and a random number generator through the roulette wheel, each follower selects one recruiter to join.

4.2.3.1 Loyalty functions

In the literature the most used decision probability is p_b^u determined by formula (4.4). From an analytical perspective it can be reasoned that p_b^u should be utilized when we want to avoid often interruptions during the backward pass. Different perspective on loyalty decision-making needs to be considered when we want to avoid behavior of the p_b^u. In recent articles [Nik13a, Nik13b] authors report that for some variants of the BCOi algorithm, a better performance might be achieved if the forward pass index (u) is omitted. Other formulas are examined in [Mak13] indicating alternative ways to decide on loyalty. New results suggest that, when dealing with combinatorial problems, introduced formulas by which probabilities are obtained are able to assist the algorithm to faster obtain optimal or near-optimal solutions. The novel formulas are: (*1*) $p_b^u = e^{-O_{\max}-O_b}$; (*2*) $p_b^u = e^{(-O_{\max}-O_b)/\sqrt{u}}$; and (*3*) $p_b^u = O_b$.

In view of a increasing number of proposals for p_b^u we introduce new term *loyalty function*, denoted by $p^{\alpha,\beta}$. It enables to review different formulas by which the values of p_b^u are determined. Particularly, it provides analytical formulation of the decision-making process that helps distinguish among different decision-making strategies. The inscription incorporates p as the function symbol in order to stay close to the common notation of the decision probability. In the superscript, variable α designates an *index of the loyalty function* ($\alpha \in \mathbb{Z}^+$). The second parameter β, know as a *class indicator*, is used to assign the function to some predefined class. Its values may be: $\emptyset, u, n_{iter}$. For example, loyalty function that describes probability $p_b^u = e^{-\frac{O_{\max}-O_b}{u}}$ is denoted as $p^{0,u}$, where value u indicates that it belongs to a specific class of functions that employ counter u. We believe that the proposed notation helps to avoid any ambiguity while addressing other probabilities within the text. Occasionally, we address the loyalty function by its index, as is the case in graphics provided in Chapters 8 and 9. It is worth noting that the loyalty function corresponds to the *loyalty criterion* introduced by Maksimović *et al.* in [Mak13].

This part of the section is exploring ten different loyalty functions employed as an integral part of an empirical study of BCO and an integral part of our research [JK16c, JK16a]. In addition to existing formulas we propose six new loyalty functions. Results of the conducted empirical study on BCO show that different loyalty functions have different impact on the performance of the BCO algorithm. Therefore, we employ graphical assessment of functions' analytical expression in order to contribute towards understanding of underlying mechanisms behind influence of loyalty decision. In particular, out goal is to elaborate in detail about the impact of functions arguments on values of probability p_b^u.

We distinguish two classes of loyalty functions:

- **Class I**: class of all functions that have one argument, a normalized value of an evaluation function, O_b.
- **Class II**: class of all functions that have two arguments, O_b and forward pass counter u or iteration counter n_{iter}.

The second class of loyalty functions is also more versatile than the first, as it may hold various combinations of two variables, while preserving values in the range between zero and one.

4.2.3.2 Loyalty function $p^{0,u} = e^{-(1-O_b)/u}$

The probability function $p^{0,u}$ is a class II function, often employed in the literature within BCO implementations while dealing with various combinatorial and continuous optimization problems. The main goal is to elevate the values of loyalty decision probabilities as the counter of forward/backward passes (u) increases during an execution of a BCO iteration. In order to observe an impact of its two arguments O_b and u on the probability p_b^u, the function was plotted for $u = 1, 2, 5, 10, 20, 50, 100$ and complete domain of O_b (i.e., $[0, 1]$). The graphics are presented in Fig. 4.10. To distinguish between different cases of u, each plot is referred to as the *level* of loyalty function. In Figure 4.10, each colored plot corresponds to the different level.

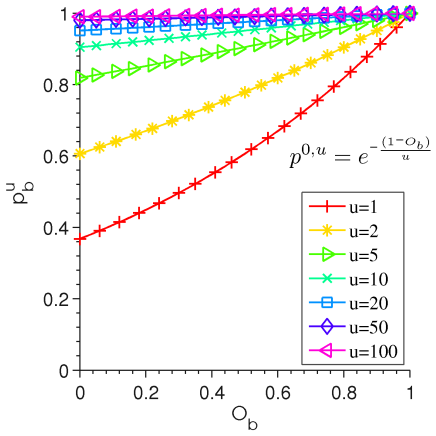


Figure 4.10: Influence of forward pass counter u and normalized value O_b on the loyalty function $p_b^{0,u} = e^{-(1-O_b)/u}$.

In Fig. 4.10 the propagation of probability values within an iteration can be nicely followed. In particular, it is interesting to contemplate the range of values for p_b^u of the worst (partial) solution. In the first foward pass, the lowest probability by which a bee decides to stay with its (partial) solution, is close to 0.36. Another interesting observation concerns plots for larger u. As the pass counter is reaching value 5, loyalty function helps establish probabilities larger than 0.8. In fact, as soon $u = 20$ it becomes almost certain that the bee will decide not to follow any of the recruiters and stay with its own solution. Bees that obtain (partial) solutions of poor quality will have a high chance to remain loyal and, consequently, transfer the solution into the next forward

pass/iteration. Namely, when p_b^u is large bees do not share the knowledge and the BCO algorithm resembles to multi-start iterative search.

4.2.3.3 Loyalty function $p^1 = e^{-(1-O_b)}$

Loyalty function $p^1 = e^{-(1-O_b)}$ was proposed by independent work of [Mak13] and [Nik13a]. It is a class I function, as it relies solely on the O_b values, therefore, characterized by only one level. Its graphical presentation is provided in Fig. 4.11. Authors of [Nik13a] were not explicit about the reason behind omission of the variable u, whereas in [Mak13] different concepts of the loyalty probability p_b^u were investigated. Without levels, the function coincides with the probability function $p^{0,1}, \forall O_b \in [0,1]$. Consequently, the lowest probability by which bees decide to remain loyal is not changing for different u, yielding values of p_b^u in the range $[0.36, 1]$ within each backward pass of the BCO algorithm. Contrary to $p^{0,u}$ that produces high number of loyal bees, the loyalty functions p^1 results in more uncommitted bees that will continue to exploit promising areas. In addition, the worst solution will be transferred into the next forward pass with probability close to 0.36, which is significantly smaller than values of $p^{0,u\geq 2}$.

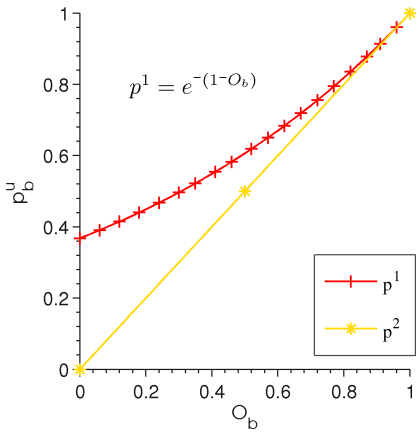


Figure 4.11: Influence of O_b on the loyalty function $p_b^1 = e^{-(1-O_b)}$.

4.2.3.4 Loyalty function $p^2 = O_b$

Loyalty function p^2 is proposed in [Mak13] as a part of investigation of various principles behind the probability decision. The function follows the expression represented as a linear function of the O_b. Graphical representation is provided in Fig. 4.11. From the graphic it is clear that the worst (partial) solution will not be transferred into the next foraging cycle, since the corresponding probability p_b^u is always equal to 0. As the values of O_b change each time after the evaluation, yielding different values of probability p_b^u, it is hard to predict their influence on the BCO performance. It seems that the function p^2 provides higher amount of intensification within the population compared against the rest of loyalty functions.

4.2.3.5 Loyalty function $p^{3,n_{iter}} = e^{-(1-O_b)/n_{iter}}$

Inspired by $p^{0,u}$ and p^1, we propose new function $p^{3,n_{iter}} = e^{-(1-O_b)/n_{iter}}$. The objective is to generate probabilities that will remain within the same interval during an execution of one iteration, however, throughout the iterations the lower bound changes. We accomplished that by replacing the u with iteration counter n_{iter}. It is a Class II function, as it depends on two arguments: O_b and n_{iter}. Its graphical presentation coincides with levels in Fig. 4.10, where instead of variable u all the levels of function $p^{3,n_{iter}}$ are defined by value of variable n_{iter}. The function is most suitable for implementations of BCO when parameter NC is not utilized, or when we want to disregard the impact of the variable u. However, as was the case for $p^{0,u}$, after n_{iter} reaches 20, probabilities p_b^u increase drastically. Therefore, another variant of the expression might be considered in the future work, such as reinitialization of the counter after it reaches some predefined threshold.

4.2.3.6 Loyalty function $p^{4,u} = e^{-(1-O_b)/\sqrt{u}}$

As suggested in paper [Mak13], in order to weaken the impact of variable u on loyalty probabilities, one can use the square root of u thus obtaining new loyalty function, here marked as $p^{4,u}$. Its graphical assessment is provided in Fig. 4.12. When $u = 1$ the graphic resembles that of other loyalty functions presented until now. Namely, the lowest point in the graphic is equal to 0.36. Unlike $p^{0,u}$, within an iteration of BCO the highest level of the $p^{4,u}$ will not converge towards 1 as fast as $p^{0,u}$. Moreover, all levels are shifted towards lower regions of the probability space indicating higher recruitment dynamics. This actually increases diversity of (partial) solutions with respect to p^1 and p^2, and increases exploitation compared to $p^{0,u}$ and $p^{3,n_{iter}}$. Assessing the plots we can conclude that, compared to $p^{0,u}$, the function provides much better balance between diversification and intensification of the search space as the number of uncommitted bees is higher for $p^{4,u>20}$. Concerning the performance of the BCO algorithm, we expect that it exhibits different results then of previous loyalty functions.

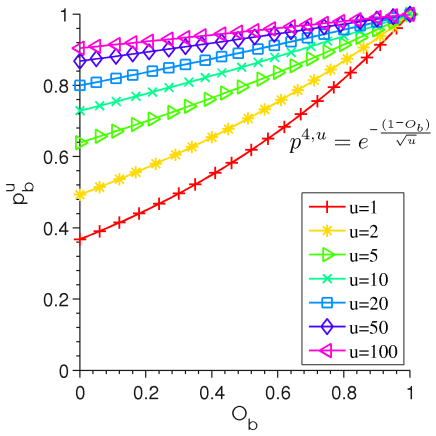


Figure 4.12: Influence of u and O_b for the loyalty function $p^{4,u} = e^{-(1-O_b)/\sqrt{u}}$.

4.2.3.7 Loyalty function $p^{5,u} = e^{-(1-O_b)\sqrt{u}/\sqrt{u+1}}$

Up to this point, all loyalty functions of argument u increase monotonically for $u \geq 1$. To reverse the idea by letting probabilities p_b^u become smaller for larger u, we propose function $p^{5,u}$. Its graphic is presented in Fig. 4.13. Its first level provides highest probability during the search for the worst (partial) solution to be transferred into next forward pass. For $u \to \infty$ levels converge towards values of p_b^1, i.e., levels are bounded by loyalty function p^1. The function $p^{5,u}$ is generated in order to avoid fast convergence towards lower regions of the probability space, however, the range within the minimal and maximal values of the worst case scenario is not as wide as first believed. The function should be used to inspire new expressions, that would allow larger range of function values.

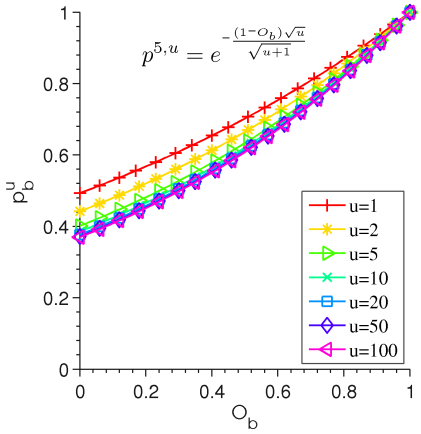


Figure 4.13: Influence of u and O_b for the loyalty function $p^{5,u} = e^{-(1-O_b)\sqrt{u}/\sqrt{u+1}}$.

4.2.3.8 Loyalty function $p^{6,u} = e^{-(1-O_b)/\ln(u+1)}$

Similar to the idea that has inspired construction of the $p^{4,u}$, instead using square root, a logarithmic function may be employed. We were able to provide new expression of loyalty function, here denoted as $p^{6,u}$ (Fig. 4.14). In this case, the influence of the variable u is even more suppressed. It is worth noting that natural logarithm is used. The graphical representation on the same domains of u and O_b is provided in Fig. 4.14. Compared to loyalty function $p^{4,u}$, levels of probabilities have decreased and the lower and upper bounds on all possible values of the loyalty function are shifted toward lower region of the probability space. The lowest point of the graphic is 0.236. Compared to $p^{4,u}$, probabilities $p^{6,u}$ are lower within each level of u.

4.2.3.9 Loyalty function $p^{7,u} = e^{-(1-O_b)/(u\ln(u+1))}$

To expand the bounds of the loyalty functions, we used $p^{7,u} = e^{-(1-O_b)/(u\ln(u+1))}$. Level graphics are provided in Fig. 4.15. By comparing Figure 4.15 with 4.14 we notice matching between the level $p_b^{7,1}$ and level $p_b^{6,1}$. This serves as an indicator that expanding toward lower regions was successful. The function $p^{7,u}$ however skips entire segments of the probability space, converging fast to high values. Namely, the probability p_b^u of the worst (partial) solution when $u = 1$, has tripled in the next level.

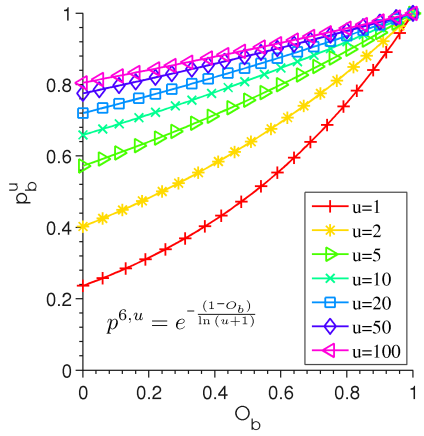


Figure 4.14: Influence of u and O_b on the loyalty function $p^{6,u} = e^{-(1-O_b)/\ln(u+1)}$.

Moreover, considering solely levels $u \geq 2$, function $p^{7,u}$ generates values larger than values of function $p_b^{0,u}$, thus causing bees to remain loyal to their (partial) solution early within an iteration.

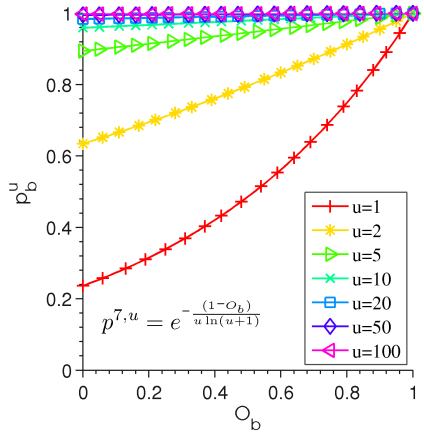


Figure 4.15: Influence of u and O_b on the loyalty function $p^{7,u} = e^{-(1-O_b)/(u\ln(u+1))}$.

4.2.3.10 Loyalty function $p^8 = e^{-2*(1-O_b)}$

Loyalty function p^8 is similar to p^1 regarding the omission of the variable u. The analytical expression of p^8 was designed in order to investigate a possible influence of coefficient in the numerator of the exponent. The graphical representation is provided in Fig. 4.16. As can be observed, the probabilities that are used in all iterations of BCO will be, for the most part, lower than the average case of $p_b^u = O_b$. Beside p^2, the function also provides the lowest point in the probability space. An additional increase in the negative power of the exponential would further expand the reaches of the loyalty functions towards the lower regions of the probability space. The coefficient should not, however, be much higher than 2.

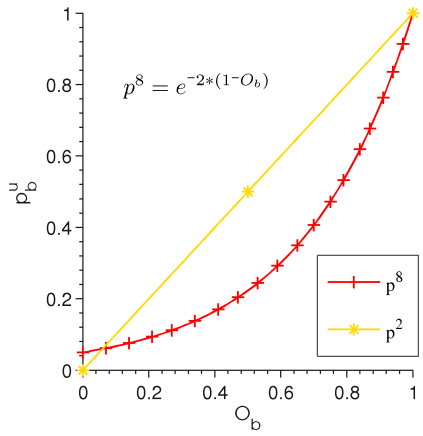


Figure 4.16: Influence of forward pass counter on the loyalty function $p^8 = e^{-2*(1-O_b)}$.

4.2.3.11 Loyalty function $p^{9,u} = e^{-(1-O_b)\ln(u+1)/\ln(u+2)}$

Similar to $p^{6,u}$, the idea is to investigate the influence of the logarithmic function on the probability space. For that reason we propose new probability function $p^{9,u} = e^{-(1-O_b)\ln(u+1)/\ln(u+2)}$. Its graphical representation is provided in Fig. 4.17. The function resembles $p_b^{5,u}$, however, when the ranges between the lowest and highest probabilities for $O_b = 0$ are compared between these two functions, $p_b^{9,u}$ shows nicer properties. Namely, the initial probabilities are higher then of $p^{5,1}$. All possible values of the loyalty functions for $u \geq 2$ are shifted toward lower regions beneath the level $p^{9,1}$. Similar to $p^{5,u}$, as u converges to higher values, the function converges towards p^1. The span between probabilities obtained when $O_b = 0$ and $u = 1, \ldots 100$ is broader then of p^5.

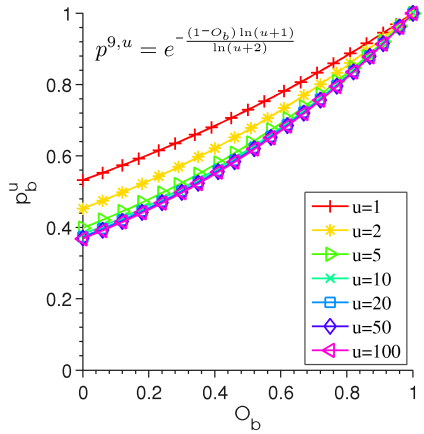


Figure 4.17: Levels of loyalty function $p^{9,u} = e^{-(1-O_b)\ln(u+1)/\ln(u+2)}$.

4.2.3.12 Models of choice

So far the concept of choice has been used in more than one occasion, which deserves more consideration. Within the BCO model two types of choices are used: those based on probabilities and the *roulette wheel*. To elaborate on the first type, let us observe formula (4.3). After the probability that the bee will stay loyal to its (partial) solution is determined, a number is chosen by uniform distribution $U(0,1]$ and compared to p_b^u. If the chosen random number is smaller than the value of p_b^u, the bee remains loyal to its solution. Otherwise, the bee becomes uncommitted, as shown in Fig. 4.18.

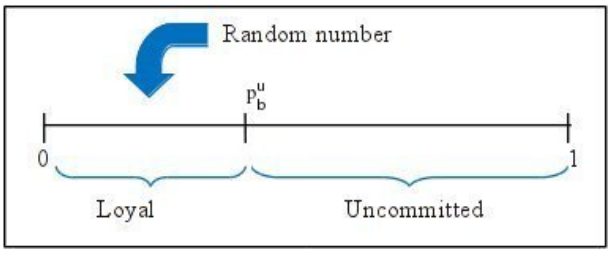


Figure 4.18: Probability based choice.

Roulette wheel represents another method of selection and it is a technique that is exploited in many domains of the theory of games. When utilizing roulette wheel all (partial) solutions are assigned with a value proportional to their quality. Therefore, the principle of the roulette wheel is to randomly select candidate solutions based of their objective (evaluation) value. Solution with better quality will have a higher chance to be selected. Equivalently, we leave the possibility that the good solution may be eliminated from the further search process as well as the possibility that the low quality solutions will be further explored.

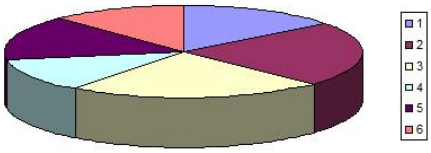


Figure 4.19: Roulette wheel.

The illustration of the roulette wheel is presented in Fig. 4.19 as the surface subdivided into wedges. The wedge represents probability associated to the corresponding (partial) candidate solution. The size of the wedge, in practice, is determined by the ratio of the corresponding normalized objective function value and the sum of the normalized objective function values for all solutions advertised by recruiters [Dav15b]. Recently, some new selection methods in BCO have been tested [Alz15]. Among roulette, tournament, rank and disruptive selection, the later one performed the best.

4.2.4 Comparing ABC, ACO and BCO

The differences between three meta-heuristics are provided in the following text. It should be noted that ABC and BCOi share most of features, whereas ABC and BCOc are different. We point out the differences between both variants of BCO and ABC as follows:

- Initial solutions are generated in various ways: randomly in ABC and in some constructive probability based manner in BCOc and or randomly in BCOi;
- Bees have different role (e.g. scouts, employed bees and onlookers) in ABC, while in BCO all bees perform the same algorithm steps;
- In ABC and BCOi the whole solutions are propagated, while in BCOc partial solutions influence the decision making process;
- In ABC the dance duration (i.e. the number of iterations a solution is propagated) is changing, contrary to BCO where only the solutions from the current iteration are propagated;
- ABC involves solution improvement, which is opposite to purely constructive basic BCO approach.

The major differences between ACO and BCO are as follows:

- The communication between ants is indirect and based on the solution components (pheromone trials), on the contrary to synchronous direct communication between bees which uses (partial) solutions;
- In the majority of the cases, ants use the local search to improve generated solutions, while the improvement is not employed concept for BCOc;
- During the solution generation process bees communicate the information from the current partial/complete solutions, while ants use the previous search experience;
- Unlike BCO, the ACO algorithms sometimes use re-initialization (of pheromone trails) to escape from local optima.

Next, we emphasize the similarities between these three methods (ABC, ACO, BCO). They are:

- The initial inspiration is coming from the foraging habits of social insects;
- All of them are population based methods;
- Artificial individuals exchange knowledge and experience during the search;
- A large number of new solutions is generated from the scratch.

4.2.5 Final remarks

The BCO method has been proved useful in dealing with hard optimization problems, mainly due to its potential to explore different solutions and by harnessing stochastic processes. The BCO methods has been successfully applied to various combinatorial and continuous optimization problems [Teo15]: routing [Luč01, Luč03a, Luč03b, Mar07, Won09, Won10a], location [Teo07, Šel08, Šel10, Dav11b, Dim11, Lev11, Soh11, Dav11b], scheduling [Dav09, Dav12, Kov13, Mou11, Ned09, Per11, Teo05, Teo08], medicine with chemistry [Sa'13, Teo13], network design [Nik13b], numeric functions optimization and mixed optimization problems [Nik13a, Sto13]. A survey on various applications of the BCO algorithm is depicted in [Teo15, Dav15a].

Beside the fact that BCO has been successfully applied to various optimization problems, enough space is left for its advancement by examining, for example, various information sharing mechanisms and different mechanisms of collaborations, or further exploring homogeneity of bees (e.g. exploiting heterogeneous bees). Preliminary results concerning heterogeneity are presented in [Nik15]. As previously discussed, different types of probability choices might be further explored, such as roulette, tournament, rank or disruptive selection. For example, the tournament selection is known as a standard for nearly all GAs [Hau04].

Theoretical analysis of the BCO was the first time tackled in [JK14a, JK14b, JK16b]. In [JK14a] the BCO algorithm convergence is analyzed in the best-so-far sense. In particular, we have determined the necessary and sufficient conditions that guarantee the convergence in probability of BCO towards an optimal solution. It is concluded that the algorithm should propagate the best-so-far solution during the search and that all feasible solutions must be reachable. In [JK14b, JK16b] the BCO method is analyzed by means of model convergence, implying a specific type of probabilistic rules that need to be implemented in order to guarantee a more efficient search. We have concluded that the important feature of the BCO algorithm is to include a learning component about the quality of previously visited solutions. As a contribution to the performance enchantment, future work might involve implementation of the theoretical results for BCO.

As a method based on the population of solutions, the BCO is suitable for application of different parallelization strategies. To the best of the authors' knowledge, prior to [Dav13] the parallel execution of BCO is treated only in [Dav11b] where independent multiple execution of different BCO algorithms is proposed and a significant speedup, while preserving solution quality, is reported. The contribution of this thesis is to investigate other various types of parallelization strategies on the BCO algorithm. The topic is to some extent covered in [Dav13] where the constructive version of BCO is implemented for the scheduling problem and tailored for distributed systems.

During the past decades an extensive work has been conducted to improve the methodological empirical analysis of meta-heuristics. The first results concerning empirical analysis of the BCO algorithm are given in [Mak13, Nik13a]. Some questions remained open, e.g., the interaction between different BCO parameters or the connection between properties of the considered problem and the BCO algorithm performance. In [JK16c, JK16a] we have addressed the question of parameter interrelation. Moreover, the topic of parameter configuration of the BCO algorithm is treated in the Chapters 7–9 of this dissertation. Future work needs to concern choice/development of tools for efficient parameter tuning based on appropriate statistical methods.

4.3 Chapter summary

This chapter provides information about the past and current development of the BCO method. We review the bees' behavioral model to help present similarities with the current BCO framework. Furthermore, we have distinguished the work of von Frish and Camazine and Sneyd that pioneered in the field of modeling of bees' foraging behavior.

In this chapter we give:

- Overview of early work towards understanding and modeling bees' behavior during foraging and the review of the first steps of the BCO model development.
- Pseudo-code of the BCO method and detailed description of the constructive and improvement variants of BCO.
- New concept of deciding on loyalty as we define *loyalty functions* and conduct a graphical assessment of dependence.
- Final remarks Brief survey of the BCO application for solving different optimization problems.
- Commentary on future development and research on BCO algorithms and applications.

Module II

Methodology and contributions

CHAPTER 5

Theoretical analysis of the asymptotic convergence of the BCO method

Theoretical analysis of a meta-heuristic algorithm may be conducted in different ways. One can observe the algorithm as Markov chains, where the algorithm's behavior is controlled by its solution steps and transition moves. Another approach is to study about the balance between diversification (exploration) and intensification (exploitation) components of the algorithm. The subject of this chapter is the theoretical analysis of the BCO algorithm properties that lead towards generation of a global optimum. In this chapter we cover formal definitions of two types of stochastic convergence used as the basic tool for theoretical analysis of various meta-heuristics. Moreover, already existing contributions to the topic are presented, while focusing on parts that are relevant for our study. We do not discuss the effectiveness of the BCO algorithm or provide improvements that would enhance the performance on any optimization problem. Rather, we provide necessary convergence conditions and prove that, when satisfied, the BCO algorithm can generate optimal solution of a considered problem if given enough time.

5.1 Motivation for theoretical analysis

Assuming that the considered optimization problem has a solution, convergence analysis is related to the question: will the desired optimum be found if the algorithm is given enough time. A serious drawback of meta-heuristic methods is the lack of information necessary to determine the actual quality of the reported solution: even if solution is optimal, no proof could be derived. Consequently, when suboptimal solution is reported we cannot estimate how far it is from some desired optimum. In the literature we can find a growing body of tutorials on theoretical analysis of convergence of meta-heuristic methods [Pin84, Liu09, Gut09, Yan11, Che13]. Nevertheless, majority of practitioners are largely focused on demonstrating, instead of analyzing method's performance. Moreover, meta-heuristics are primarily investigated experimentally, for the most part associated with their concrete engagement and implementations.

The difficulty in theoretical analysis of meta-heuristic methods is generally a consequence of highly nonlinear, complex, and stochastic interactions between their various components [Yan11]. There are different approaches to theoretical analysis of stochastic algorithms, such as observing the processes as dynamical systems and Markov chains and using tools of computational complexity theory and probability theory [Yan14]. For example, finite state Markov chains is a common approach to model EA algorithms to

compute the optimal convergence probability with finite search sets [Ras11]. [Rud94] investigate convergence properties of the canonical GA by means of Markov model, and shown that only variants of CGA with an elite reserve policy converge to a global optimum. [Rud96] provides conditions under which EA algorithms with an elitist selection rules converge to the global optimum. Among mentioned methodologies, probability theory is the most common approach for analysis of non-deterministic algorithms. A good review of this subject is given in [Che13].

Intensive applications of meta-heuristics and their remarkable success in solving various optimization problems has initiated the development of a mathematical methodology for explaining and predicting a performance of meta-heuristic algorithms [Pap98]. The methodology depicts formal aspects, e.g., convergence and scalability analysis, the design of novel operators, universal computation and approximation capabilities, hybridization with other methodologies and approaches, the automatic parameter tuning, etc. [Cas07]. Some aspects of an algorithm's behavior are in theory well explained (e.g., convergence analysis and scalability of ACO and PSO), while others remained unanswered (e.g., connection between convergence speed and properties of a problem) [Cas07, Gut11]. It can be argued that the lack of theory can influence a number of meta-heuristic prospects, such as the development of more effective meta-heuristic algorithms, identification of the meta-heuristic the most appropriate for a given problem, more thorough investigations into the benefits and/or drawbacks of particular meta-heuristic, etc. [Glo86, Hoo95, Pap98, Wat10].

Theoretical analysis of meta-heuristics is commonly addressing a question of convergence: will a given meta-heuristic method find an (global) optimal solution when given enough resources [Dor05]. The question about the asymptotic convergence dates back to the beginnings of the random search method and analysis of its properties. Random search method was first considered in the design of experiments for the purpose of seeking maxima. According to [Whi70] the *pure random search* is mentioned in [And53] and later elaborated in [Bro58]. The method is based on selecting n random points and taking the one with the largest value of an optimal function f. In order to study the convergence of a random search method, Rastrigin in [Ras63] observes *creeping random search*. The method is based on improving the current solution by conducting step moves. In the same paper the author considers convergence rate and compares the algorithm with the steepest decent method. Other researchers, such as Karnopp in [Kar63], have stressed the importance of the random search methods for general optimization problems. However, the general convergence proofs were presented by Solis and Wets [Sol81], almost 20 years later, for two versions of the *conceptual* (generic) random search algorithm. Namely, the authors distinguished local and global versions. Opposed to local search, the algorithm is considered as a global search method if, for any $A \subset S$, the probability of repeatedly missing the set A must be zero [Sol81].

Theoretical verification of meta-heuristic convergence is a topic of many research papers: SA [Haj88, Gra94, Ste00], GHC (generalized hill-climbing) [Sul01, Joh02a, Jac04b, Jac04a], PSO [Tre03, Zen04, VDB06, Jia07, GG12], CE (cross-entropy) [Mar05], GA [Har90, Rud94, Sch01], ACO [Gut00, Stü02, Gut02, Zlo04, Gut11, Köt12], TS [Han01] and VNS [Bri04]. Various approaches can be distinguished. Difference between them is that, for the most part, the starting point is a specific algorithm not related to some other meta-heuristic method. For this reason authors of [Zlo04] and [Gut09, Gut10] provided a general framework to encompass most (or all) known meta-

heuristics.

5.2 Instance- and model-based algorithms

An alternative framework, one that would enable improving the performance of majority of meta-heuristic algorithms, were proposed in [Zlo04, Gut09]. These frameworks are based on analyzing the parameters of the corresponding meta-heuristic method. The authors discuss two main approaches to constructing meta-heuristic methods. In particular, borrowing a notation from the machine learning field, Zlochin *et al.* [Zlo04] proposed two types of heuristic algorithms: *instance-based* and *model-based*. An instance-based algorithm is one that uses only current solution or a set of current solutions, in order to generate new candidate solutions. Model-based algorithms (MBS) are algorithms that generate candidate solutions using a parameterized probabilistic model. The authors of [Zlo04, Gut09] agree that, in order to generate high-quality solutions, the considered meta-heuristic has to utilize previously attained knowledge. This actually means that the meta-heuristic method has to learn from previously visited solutions how to concentrate its search on the more promising areas of the solution space, i.e., the regions containing solutions of higher quality.

According to [Zlo04], a meta-heuristic method satisfies the *model-based search properties* if it attempts to solve the optimization problem by repeating the following two steps:

- Candidate solutions are constructed using some parameterized probabilistic model.
- The candidate solutions are used to modify the model in such a way to concentrate the search toward more promising regions (containing solutions of better quality).

For that type of meta-heuristics the *model-based parameter scheme* was adopted as an assurance for *model convergence* by Gutjahr [Gut09]. Then, the considered meta-heuristic can be analyzed with respect to its parameters.

Important component of the model requires an *update rule* for the method's parameters and/or structure [Zlo02, Zlo04]. A role of the update rule is to define a probability vector that will explicitly keep gathered knowledge (statistics). On the contrary, in the classical algorithm designs, to implicitly maintain statistics about a search space (about information that is carried into the next iteration) according to [Bal95] a population of solutions is commonly used. To combine the best from both worlds, in [Bal95] the authors proposed an abstract genetic algorithm, so called population-based incremental learning (PBIL), where the classical GA operators were replaced by three parameters: (a) a number of samples, (b) a learning rate, and (c) a number of probability vectors. Authors have compared PBIL and GA and showed that PBIL outperforms GA on a four peaks problem, which they also introduced in the same paper. Incorporating update rules inspired a new design of algorithms (with good grounds for theoretical studies) and development of more sophisticated algorithms [Ras11]. In particular, the early work on PBIL motivated the model-based approach in [Zlo02, Zlo04].

To assure model convergence of the meta-heuristic Gutjahr presented *generic algorithm*, which we briefly review here. First, let us consider an optimization problem that requires minimization of a real-valued function f on a search space S. Generic

algorithm is an iterative algorithm that uses two structures: m_t and L_t, where t is an iteration index. In iteration t structure m_t defines a state of memory, while L_t represents a list of solutions ($\boldsymbol{x}_i \in S$) employed as *sample points* in iteration t. The description of the algorithm consists of following steps:

1. Initialization of memory m_1 by specified rules;
2. $t \leftarrow 1$;
3. Until stopping criterion is satisfied:
   a) Determine the list L_t as a function $g(m_t, \xi_t)$ of memory state m_t and a random influence ξ_t;
   b) Determine a value of objective function $f(\boldsymbol{x}_i)$ for all $\boldsymbol{x}_i \in L_t$, and generate a list L_t^+ of pairs $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$;
   c) Determine new memory state m_{t+1} as a function $h(m_t, L_t^+, \xi_t')$ of current memory state m_t, list of solution-value pairs L_t^+ and random influence ξ_t';
   d) $t \leftarrow t + 1$.

Structure m_t consists of two components: m_t^s, part of memory that contains all solution-values (*sample-generating part*) and m_t^r, part of memory that contains reports (*reporting part*). The component m_t^s holds all the information required by function g to generate the list L_t of sample points. The component m_t^r accommodates all other relevant information, for example, solution of the highest quality found so far, namely *best-so-far*, $\boldsymbol{x}^{bsf}$. In the initialization phase, a value of variable $\boldsymbol{x}^{bsf}$ is set arbitrary until, in some iteration t, it copies a value from $\hat{\boldsymbol{x}}_t$, the best among $\boldsymbol{x}_i$ solutions that satisfies the condition $f(\hat{\boldsymbol{x}}_t) < f(\boldsymbol{x}^{bsf})$. Furthermore, function $g(m_t, \xi_t)$ specifies a probability distribution for new elements of list L_t, whereas function $h(m_t, L_t^+, \xi_t')$ denotes probability distribution of the new state of memory m_{t+1}. It is important to note that information about the problem instance can also be used as an argument of functions g and h. In addition, we want to emphasize that the best-so-far solution in iteration t is used as a current approximation of a global optimal solution [Gut09].

Gutjahr [Gut09] provided several examples for generic algorithms of meta-heuristic method, e.g., the GHC method, canonical GA (CGA) and ACO. Here, we review a procedure for ACO. Generic ACO uses procedure structures: a sample part m_t^s, containing a matrix of real-valued pheromones, and m_t^r containing $\boldsymbol{x}^{bsf}$, possibly also an iteration-best $\hat{\boldsymbol{x}}_t$. Furthermore, the list L_t contains k solutions. The procedure steps for ACO are: (1) let k agents construct k random solutions, where the moves are governed by the pheromone values stored in m_t^s and thus obtain L_t; (2) update m_t to m_{t+1} by: (a) update $\boldsymbol{x}^{bsf}$ based on k new solutions in L_t^+, and (b) apply specific *pheromone update rule* to obtain new pheromone values from the current values by reinforcing the components of the solutions contained in $\boldsymbol{x}^{bsf}$. Similarly, generic algorithm for BCO method can be deduced, which we present in section 5.4.

5.3 Theoretical background

The main theoretical contribution of this thesis (Section 5.4) concerns the proof of convergence of the BCO method towards an optimal solution. Therefore, we recall

some of the basic tools of probability theory and establish the notation necessary to present our results.

Most of the meta-heuristics are stochastic search algorithms, and therefore, in order to obtain formal definitions of algorithm convergence, tools of probability theory are used while we consider the series of random variables with a common distribution. Generally, the random variables are not independent. For meta-heuristics the independence holds only in case of pure random-walk search techniques. Two important definitions of stochastic convergence are provided in the following [Bil86, Pap91c].

From mathematical definition, an expression *convergence* is defined to answer a question whether or not a series of elements $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t, \ldots)$, $\boldsymbol{x}_t \in \mathcal{X}$, converges to a fixed value $\boldsymbol{x}^*$. We also define some distance function d between these elements. The sequence $(\boldsymbol{x}_t)_{t=1}^{\infty}$ converges to a fixed value $\boldsymbol{x}^*$, if for each $\varepsilon > 0$, there is an integer t_0 such that $d(\boldsymbol{x}_t, \boldsymbol{x}^*) < \varepsilon$ for all $t \geq t_0$, where $d(\cdot)$ is a convenient metric function (for vector it is usually a vector norm $||\cdot||$). If the space $\mathcal{X}$ is finite, the simplified version of the definition is applicable: $\boldsymbol{x}_t$ converges to $\boldsymbol{x}^*$ if and only if there is some t_0 such that $\boldsymbol{x}_t = \boldsymbol{x}^*$ for all $t \geq t_0$.

Let $(X_1, X_2, \ldots)$ be a series of random variables with the common distribution.

Definition 8 (convergence in probability). A sequence of random variables $(X_1, X_2, \ldots)$ converges *in probability* to a random variable X^*, if for all $\varepsilon > 0$,

$$\lim_{t\to\infty} \Pr(\{d(X_t, X^*) \geq \varepsilon\}) = 0$$

where d is the distance function on the space $\mathcal{X}$ from which the random variables X_t take their values. ◇

Definition 9 (convergence with probability one). A sequence of random variables $(X_1, X_2, \ldots)$ converges *with probability one* (short: w. pr. 1) or *almost surely* to a random variable X^*, if

$$\Pr(\{\lim_{t\to\infty} X_t = X^*\}) = \Pr(\{X_t \to X^*\}) = 1,$$

i.e., if with probability one the realization $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots)$ of the sequence $\{X_t\}_{t=1}^{\infty}$ converges to the realization $\boldsymbol{x}^*$ of X^*. ◇

For random variables that satisfy condition of Def. 8 it is said that the weak law of large numbers holds uniformly on $\mathcal{X}$ [Sha96]. Usually the definitions are specialized to the case where X^* takes a constant element $\boldsymbol{x}^*$. In this case, if $\mathcal{X}$ is a finite set, convergence of X_t to $\boldsymbol{x}^*$ w. pr. 1 holds exactly if

$$\Pr(\{\exists u \geq 1 : \; X_t = \boldsymbol{x}^* \quad \forall t \geq u\}) = 1,$$

and convergence of X_t in probability holds exactly if

$$\lim_{t\to\infty} \Pr(\{X_t = \boldsymbol{x}^*\}) = 1.$$

However, the definition of the convergence in probability can be generalized by saying that if set of limit solutions, denoted as $\mathcal{X}^*$, is the subset of $\mathcal{X}$ then

$$\lim_{t\to\infty} \Pr(\{X_t \in \mathcal{X}^*\}) = 1.$$

Moreover, it is important to provide the most often used elaboration of the convergence in probability in the literature: "*The basic idea behind the convergence in probability is that the probability of an **unusal** outcome becomes smaller and smaller as the sequence progresses*". What is referred by an "unusal" event is that $d(X_t, X^*)$ exceeds any prescribed, positive value ε [Kar93].

Considering meta-heuristics, *convergence* is defined to answer a question whether or not current solution, proposed by a meta-heuristic method, converge to an optimal solution and, if yes, how fast this happens [Gut09]. In particular, the sequence of random variables corresponds to iterations of a meta-heuristic algorithm. At the end of each iteration, the algorithm reports a solution. To establish analogy with previous definitions, we denote any solution found in iteration t as $\boldsymbol{x}_t = (x_t^1, x_t^2, \ldots, x_t^n)$, where $x_t^i \in \mathbb{R}$ is the element (component) of the solution. In case of population-based algorithms, at the end of a iteration t the algorithm reports the *iteration-best* solution $\hat{\boldsymbol{x}}$. Let us presume that the algorithm incorporates global knowledge. Thus, we obtain "best-so-far" solution in iteration t, denoted by $\boldsymbol{x}_t^{bsf}$. Therefore, analyzed sequence $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t, \ldots)$, where $\boldsymbol{x}_t \in \mathcal{X}$, denotes a sequence of "best-so-far" solutions $\boldsymbol{x}_t^{bsf}$ generated by the meta-heuristic method after t iterations. Let global optimum be denoted as $\boldsymbol{x}^*$. The distance function d is defined as $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = |f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)|$, i.e., as the absolute difference between the corresponding values of the objective function. For most of the meta-heuristic methods it holds that once an optimal solution is reached, it is propagated across the forthcoming iterations. Consecutively, our sequence becomes $(\boldsymbol{x}_1^{bsf}, \boldsymbol{x}_2^{bsf}, \ldots, \boldsymbol{x}_t^{bsf}, \ldots, \boldsymbol{x}^*, \boldsymbol{x}^*, \ldots)$.

Dorigo and Blum recognized two types of convergence [Dor05]: *convergence in value* and *convergence in solution*. Goal of convergence in value is to evaluate the probability of finding an optimal solution at least once. However, when investigating the convergence in solution, the aim is to evaluate the probability that the algorithm reaches a state in which it generates optimal solution. [Gut09] identifies the two types of convergence as the *best-so-far convergence* and more sophisticated *model convergence*. Unlike best-so-far, the model convergence is more complicated to prove since the compromise between exploration and exploitation of the search space needs to be balanced by properly adjusting algorithm's parameters. In addition, proof of model convergence is focused on the m_t^s part of the memory.

Slightly different framework for algorithm analysis in terms of Markov chain Monte Carlo was formulated in [Yan11]. The convergence analysis of SA and PSO was examined in this context as well as the analysis of a newly proposed method, the firefly algorithm. In this thesis, however, we endorse the approach proposed in [Gut09].

5.3.1 Best-so-far convergence

The first type of convergence refers to a question whether or not the best-so-far solution $\boldsymbol{x}_t^{bsf}$ converges to some optimal solution $\boldsymbol{x}^*$, as $t \to \infty$. Considering meta-heuristics, this question actually refers to the convergence of the objective function values $f(\boldsymbol{x}_t^{bsf})$, $(t = 1, 2, \ldots)$ to the optimal function value $f(\boldsymbol{x}^*)$. Thus, we may ask under which conditions it can be guaranteed that $f(\boldsymbol{x}_t^{bsf})$ converges ("w. pr. 1" or "in probability") to $f^* = \min\{f(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}\}$.

Proving the best-so-far convergence is generally simple. The concept originates from theoretical work of random search method, e.g., in [Sol81]. [Sol81] derive general

convergence conditions for global and local search versions of the conceptual random search algorithm, for problem of minimization of function f on $S \subset \mathbb{R}$. The conceptual algorithm is based on step $x^{t+1} = D(x^t, \xi^t)$, i.e., mapping of solution x^t and random samples ξ^t onto a new solution. This means that x^{t+1} is generated according to some conditional probability measure, denoted by μ_t. We focus on the global search variant and the two provided conditions. The first one is that for map D with domain $S \times \mathbb{R}^n$ the condition $f(D(x,\xi)) \leq f(x)$ must hold. The second condition is that for any (Borel) set $A \subset S$ with positive Lebesque measure $v(A) > 0$, following must hold: $\prod_{t=0}^{\infty}[1 - \mu_t(A)] = 0$. The later means that the probability to repeatedly miss set A, over all iterations, must be zero [Sol81]. By means of Markov chain theory [Rud96] concludes similarly for EA algorithms, i.e., regardless of the starting point, a nonzero probability to visit any subset of $\mathcal{X}$ needs to be enabled. The both requirements from [Sol81] can be substituted by only one condition, that the probability (p^*) of hitting an optimal solution in a single iteration of the search procedure is strictly greater than zero [Gut09]. The conceptual algorithm differs from the pure random search method as it incorporates some view of knowledge. Namely, generic algorithm of the pure random search relies on the reporting part of the memory m_t^r that consists only of $\boldsymbol{x}_t^{bsf}$, while the sample-generating part m_t^s is empty. Because m_t^s does not hold any information, the choice of a new solution is performed independently from previous iterations [Gut09], whereas conceptual random search algorithm uses m_t^s when choosing μ_t.

Best-so-far convergence has been proven in [Har90, Rud94] for certain variants of GA, transferring "elite" solutions from generation to generation (*elitist selection* implies that the best individuals survive with probability one). Rudolph [Rud94] proved that even though original CGA never converges to the global optimum for any selection of the initial population, crossover operator, and objective function, while modified versions, like elitist GA, do. In [Har90] the author concluded that global convergence of simple GA can be achieved not only if the best individuals survive, but also (in a spirit of simulated annealing) in the cases when some randomly selected individuals are transferred to the next generation. The various stochastic convergence properties for EA were summarized recently in [Rud12]. The best-so-far convergence has also been proven for some variants of ACO [Stü02] and PSO [Zen04].

Among the single solution based methods, the best-so-far convergence was first proven for SA [Haj88]. This and the later convergence results for SA [Gra94, Ste00] serve as the initial point in analyzing the convergence of many other methods, both population and single solution based. In [Han01] the author proposed a Convergent TS (CTS) algorithm with the special rules to deal with the situations where cycling is unavoidable. The best-so-far convergence of VNS and MLS was considered in [Bri04]. In the case of MLS, it was concluded that the probability of obtaining a local solution in any iteration depends on the volume of the number of points in the so called entrapment zone, which leads to exponentially increasing the time for finding a global solution. The necessary condition for the convergence of VNS was expressed with respect to the parameter k_{max} defining the largest neighborhood visited: it has to be sufficiently large to cover the entire search space S.

The convergence concept described above may look nice. However, this concept is too "generous" since it turns out that even very inefficient search algorithms converge to the optimum. The standard example of this observation is the random search. Furthermore, it turns out that the expected value of the first hitting time $E(T_1)$, is usually

very large which makes this convergence concept unhelpful for the practice. The first hitting time is given as $T_1 = \inf\{t : f(\boldsymbol{x}_t^{bsf}) = f(\boldsymbol{x}^*)\}$. The measure is typically used in the runtime analysis of stochastic algorithms that deals with questions regarding computational effort needed to obtain an optimal solution for the first time [Gut10].

5.3.2 Model convergence

To ensure that provable convergence properties correspond also to a good runtime behavior, the best-so-far convergence is not enough. In addition, we need to address the issue of a poor performance as the search sharply decreases in the neighborhood of the optimal solution [Pin84]. In [Gut09, Zlo04] it was stated that an efficient meta-heuristic method should concentrate the search more and more on the most promising areas of the search space S exploiting the previous search experience. The current set of solutions was referred to as the current *model* for the search distribution [Gut09]. In the model-based view, the generation of new search points depends on the current model. The newly obtained objective function values are evaluated, and the obtained information is then fed back for a modification of the model.

By the argumentation above, a runtime behavior superior to that of random search can be expected if it can be shown that the model (the set of current solutions) converges, as $t \to \infty$, to some state that supports only the generation of optimal or at least high quality solutions. This type of convergence is actually *model convergence* [Gut09].

Contrary to the proofs of best-so-far convergence which are quite easy, model convergence proofs have to take the exploration/exploitation tradeoff explicitly into account and only succeed under parameter assumptions ensuring a proper balance between these two factors. The main conclusion in [Gut09, Zlo04] is that parameter values have to change during the search. Typically, the convergence results lead to rather narrow conditions for parameter schemes within which model convergence holds; outside the balanced regime, either a surplus of exploitation yields premature convergence to a suboptimal solution, or a surplus of exploration produces random-search-type behavior without model convergence (although best-so-far convergence may hold).

Historically, the first model convergence results have been found in the SA field [Haj88]. In order to implement convergent SA one has to change the cooling temperature rate $T(t)$ as a non-increasing function of the iteration number t that approaches zero as $t \to \infty$. Hajek [Haj88] suggested the following function for $T(t)$:

$$T(t) = \frac{c}{\log(t+1)}$$

for suitably selected constant c. Furthermore, model convergence results have been provided for GHC algorithms [Joh02a, Jac04b, Jac04a]. The authors work already on a general level and present rules applicable to wide range of meta-heuristic methods.

Results for model convergence exist also for some ACO variants [Gut00, Gut02]. For a Graph-based Ant System (GBAS), two cases were identified [Gut02]:

1. For GBAS with time-dependent evaporation rate $\rho(t)$ the convergence conditions are as follows

$$\rho(t) \leq 1 - \frac{\log t}{\log(t+1)}, \ (t \geq t_0) \text{ for some } t_0 \geq 1,$$

and

$$\sum_{t=1}^{+\infty} \rho(t) = +\infty.$$

2. GBAS with time-dependent pheromone lower bound $\tau_{min}(t)$ converges if the following condition is satisfied

$$\tau_{min}(t) = \frac{c(t)}{\log(t+1)} \text{ for } t \geq 1 \text{ with } \lim_{t \to +\infty} c(t) > 0.$$

In order to design GA as a model-based search, various authors proposed either to use time-dependent mutation and/or crossover probabilities or to substitute classical GA operators by some other, ACO-based probabilistic transformations. The brief overview of these ideas can be found in [Sch01, Zlo04]. The model convergence has also been considered for a variant of Cross Entropy Optimization [Mar05].

Detailed analysis of the principles incorporated in the PSO algorithm described in [GG12, Jia07] adopted completely different approach to the definition of convergence properties for this meta-heuristic method. Starting with the rules for updating positions and velocities of each particle in each iteration, the authors identified three influencing parameters (the inertia weight, global and local accelerations). Since their update rules represent the motion equations involving differentiation, the values for the influential parameters have to be chosen in such a way as to guarantee the stability of the resulting equations. This stability is, however, closely related to the convergence properties of the PSO meta-heuristic. In [GG12] the authors presented a taxonomy of various PSO-based algorithms and pointed out the appropriate values of the influential parameters for some of them.

5.4 Convergence analysis of the BCO method

Each implementation of the BCO algorithm is tailored for a particular optimization problem that one wants to solve. This involves the solution representation, the rules for constructing/modifying current solutions, as well as the evaluation and the comparison of these solutions. Once the program is completed, it is executed on each of the given instances until the stopping criterion is satisfied.

The final solution of the BCO execution is identified as the best solution found before the stopping criterion is fulfilled ($\boldsymbol{x}^{bsf}$). If the optimal solution for a particular problem instance is not known, we cannot discuss the quality of the final BCO solution. Is it the optimal one or, if not, how far is it from the desired optimum? The only thing we can do is to increase the maximum number of iterations and, possibly, obtain a better final solution. A numerous successful applications of the BCO method illustrated its efficiency in an experimental way. Moreover, there are some recent papers dealing with the empirical evaluation and parameter calibration of BCO [Nik13a, Mak13, JK16c, JK16a]. The main goal of this thesis is to provide some results towards theoretical analysis of BCO.

For the considerations to follow, we assume that the optimization problem involves minimization of some objective function $f(x)$ on the feasible space $\mathcal{X}$. Additionally, the stopping criterion is defined as the maximum number of iterations. For the course of

our analysis we borrow the notation from [Jac04b, Gut09] and define the following events.

Definition 10. Let $\boldsymbol{x}$ be any feasible solution of the considered problem. For each iteration $t \geq 1$:

- $C(t)$ represents the event that $\boldsymbol{x}_t = \boldsymbol{x}$, i.e., the considered solution was generated in iteration t. The complementary event is denoted by $C^c(t)$.
- $B(t)$ denotes the event $C^c(1) \cap C^c(2) \cap \cdots \cap C^c(t)$, i.e., the algorithm does not visit $\boldsymbol{x}$ over first t iterations. $B^c(t)$ is the complementary event to $B(t)$.
- $B = \bigcap_{t=1}^{\infty} B(t)$ describes the event that $\boldsymbol{x}$ cannot be generated by the algorithm, i.e., no iteration at all produces the consdidered solution $\boldsymbol{x}$.
- $r(t) = \Pr(B^c(t)|B(t-1)) = \Pr(C(t)|B(t-1))$ is the probability that in the iteration t, solution $\boldsymbol{x}$ is produced, although it has not yet been produced in any of the previous iterations. ◇

The events in Def. 10 might specify different outcomes regarding the type of the solution. In particular, we utilize the same notation for events related to the optimal solution. Thus, before utilization of Def. 10 in the thesis we always specify which type of solution is being observed.

An important feature of the event $B(t)$ is the type of the sequence it produces. Let $\boldsymbol{x} = \boldsymbol{x}^*$ and $B(1)$ denote the outcome that the algorithm did not produce an optimal solution in the first iteration. If Ω denotes the space of all possible outcomes, then $B(1) = \Omega \backslash C(1)$. Consequently, the $\{B(t)\}_{t=1}^{\infty}$ is a non-increasing sequence of events[(1)], i.e.,:

$$B(1) \supseteq B(2) \supseteq \cdots \supseteq B(t) \supseteq B(t+1) \supseteq \cdots .$$

5.4.1 Approximation of an optimal solution

To establish solid grounds for convergence proof of population-based method, we need to consider more carefully how the sequence $(x_t)_{t\in\mathbb{N}}$, introduced in Section 5.3, is being constructed.

Let $\{\boldsymbol{x}_t^1, \boldsymbol{x}_t^2, \ldots, \boldsymbol{x}_t^b, \ldots, \boldsymbol{x}_t^B\}$ denote a collection of solutions generated in iteration t, where each solution consists of n elements. The best solution in iteration t with regard to some objective function $f(x)$ of the minimization problem, is denoted as $\hat{\boldsymbol{x}}_t$ such that $f(\hat{\boldsymbol{x}}_t) = \min_{b\in\{1,\ldots,B\}} f(\boldsymbol{x}_t^b)$. The BCO algorithm can start with some initial value for $\boldsymbol{x}_1^{bsf}$, after which it is being calculated as:

$$\boldsymbol{x}_{t+1}^{bsf} = \begin{cases} \hat{\boldsymbol{x}}_t & \text{if } f(\hat{\boldsymbol{x}}_t) < f(\boldsymbol{x}_t^{bsf}); \\ \boldsymbol{x}_t^{bsf} & \text{otherwise} \end{cases} \tag{5.1}$$

for $t = 1, 2, \ldots$. In other words, the best-so-far solution is updated only if better solution is produced. Therefore, the update rule (5.1) does not forbid generation of other solutions from $\mathcal{X}$. Practicality of this approach is in observation of one instead

[(1)] In [Gut09] the author mistakenly reported that the sequence $\{B(t)\}_{t=1}^{\infty}$ is non-decreasing.

of population of solutions. Moreover, the rule can be easily defined for a maximization problem.

A good approximation of the optimal solution would be a solution that is not far away from the optimum and/or an optimum are reachable from any current best-so-far solution. Therefore, the best-so-far solution is used as a current approximation of the global optimal solution in iteration t [Gut09]. In order to analyze the convergence of BCO we observe the sequence of best-so-far solutions, i.e., $(\boldsymbol{x}_1^{bsf}, \boldsymbol{x}_2^{bsf}, \ldots, \boldsymbol{x}_t^{bsf}, \ldots)$, i.e., their corresponding objective function values, where one solution can be propagated through more then one iteration. To assure global convergence of BCO it is important that either all solutions can be reached from any initial point or restart procedures are used to avoid being trapped in a local minimum (from which optimal solution cannot be generated).

5.4.2 Generic BCO algorithm

To apply general set of rules that helps establish convergence properties of a BCO method we need to assure that it belongs to a collection of meta-heuristic methods that can be depicted under a single framework proposed by Gutjahr [Gut09]. The generic procedure for the BCO method contains the following steps:

1. Perform NC forward/backward passes and obtain the list L_t as a function of memory state m_t and a random influence.

2. Evaluate all solutions and generate L_t^+.

3. Determine new memory state m_{t+1} based on L_t^+, m_t and some random influence by updating $\boldsymbol{x}_t^{bsf}$ and evaluation model of possible moves (to be used in step (2)(i)(a) of the pseudo-code presented in Fig. 4.4, pg. 60).

The above steps describe only one iteration of the BCO algorithm to avoid stating obvious parts of the initialization and the stopping criteria condition. The generic BCO algorithm implies both BCOc and BCOi algorithms. However, it does not specifies their differences, which may influence convergence properties.

5.4.3 Various cases of the BCO algorithms

To summarize the differences between constructive and improvement BCO we point out that BCOc operates on partial solutions, adds components[(2)] to them and makes loyalty and recruitment decisions that depend on the incomplete knowledge. For example, the current (partial) value of the objective function or the lower/upper bound for the final solution quality guide the search process. On the other hand, BCOi uses the complete solutions all the time, tries to improve them during the execution of each forward pass and decides on loyalty and recruitment based on the complete information related to the quality of current solutions.

[(2)] To avoid any ambiguity, in this chapter we single out the term *component* to refer to a component of the problem, e.g., a city in TSP.

In addition to the described two classes of the BCO algorithm, from the mathematical verification point of view we need to introduce two more classes: The variants composed of mutually independent iterations (either constructive [Teo05, Teo06, Teo07, Mar07, Teo08, Eda08, Šel10] or improvement [Dav11a]) and the variants involving the transmission of some global knowledge through the iterations (again either in constructive [Dav12] or improvement [Nik13a, Nik13b] case). The main differences between these two will be pointed out later in this section.

5.5 Best-so-far convergence of BCO

We first identify the sufficient condition assuring that an optimal solution can be generated by any bee when the number of iterations is large enough. Then, we show that the current best solution generated by BCO converges to one of the optimal solutions, as the number of iterations increases, with the probability one. More precisely, we prove the best-so-far convergence of the BCO algorithm.

As we already mentioned, two types of the BCO implementations should be distinguished: The one consisting of independent iterations and the other with global knowledge exchanged between iterations. In order to make our BCO able to generate an optimal solution we need to assure that the whole search space is accessible, i.e., that any solution can be generated by at least one bee with the strictly positive probability ($p > 0$). If the iterations are independent, then the above mentioned probability does not change during the algorithm execution. Even in this case the current best solution $\boldsymbol{x}_t^{bsf}$ is kept and always included in the sequence of the "best-so-far" solutions, defining the final result of BCO at the termination condition. Essentially, this reflects the assumption that the Markovian chain described by the search process is irreducible [Har90]. The iteration independency means that $\boldsymbol{x}_t^{bsf}$ does not influence the search process during the next iteration. In this case BCO acts more or less as a population based search of "random walk" type. However, as it was stated in [Gut09], the best-so-far convergence holds and we prove it for the BCO algorithm here.

Theorem 5.1. (Independent case). *Let $P^*(t)$ be the probability that the BCO algorithm generates an optimal solution $\boldsymbol{x}^*$ at least once within the first t iterations. Let p^* be the probability that any bee generates the optimal solution. Then for an arbitrary $\varepsilon > 0$ and for a sufficiently large t, $p^* > 0$ implies $P^*(t) \geq 1 - \varepsilon$. Asymptotically, this yields*

$$\lim_{t \to +\infty} P^*(t) = 1.$$

Proof: ($\Rightarrow$)

By Def. 10 and assuming that $\boldsymbol{x}$ represents $\boldsymbol{x}^*$, we have

$$\begin{aligned} P^*(t) &= 1 - \Pr(\{\boldsymbol{x}^* \text{ is not found in any of } t \text{ iterations}\}) \\ &= 1 - \Pr(B(t)) \\ &= 1 - \Pr(C^c(1) \cap C^c(2) \cap \cdots \cap C^c(t)). \end{aligned} \tag{5.2}$$

If the iterations are independent then

$$\begin{aligned} P^*(t) &= 1 - \Pr(C^c(1)) \cdot \Pr(C^c(2)) \cdot \ldots \cdot \Pr(C^c(t)) \\ &= 1 - \prod_{i=1}^{t} (1 - p^*) = 1 - (1 - p^*)^t. \end{aligned} \tag{5.3}$$

Asymptotically, this yields

$$1 \geq \lim_{t \to +\infty} P^*(t) = 1 - \lim_{t \to +\infty} (1 - p^*)^t = 1.$$

($\Leftarrow$) Suppose that $\boldsymbol{x}^*$ is found by BCO. This implies $p^* > 0$. ∎

When the combinatorial optimization problem is considered the search space is discrete (usually also finite), and therefore convergence means that starting with some t_0 it holds that $\boldsymbol{x}_t^{bsf} = \boldsymbol{x}^*$ for all $t \geq t_0$. Consecutively, we can easily prove the converse of Theorem 5.1, i.e., that the best-so-far convergence of BCO implies $p^* > 0$. Indeed, once the optimal solution is found by BCO, it is obvious that the probability of its generation must be strictly positive.

Therefore, in combinatorial optimization case the necessary and sufficient condition for an implementation of the BCO algorithm to be convergent is that selected solution representation and construction/modification rules allow the generation of an optimal solution with the probability strictly larger than 0. For example, in solving TSP if the solution is represented as a permutation of cities, than the probability to generate an optimal tour equals at least $1/n! > 0$. In solving the p-median or p-center problem, to obtain the optimal solution a bee has to select appropriate p among n locations. The number of all selections of p elements out of the given n is $\binom{n}{p}$, and therefore, $p^* \geq 1/\binom{n}{p} > 0$.

In the case when iterations are not independent, the probability that any bee generates the optimal solution is not constant anymore. It depends on the iteration number and the collected knowledge, i.e., we can denote $p^* = p^*(t)$. The necessary condition for best-so-far convergence is the existence of strictly positive lower bound p_{min} for $p^*(t)$. This is expressed in the next theorem. The proof is inspired by proof of Lemma 1 in [Jac04b].

Theorem 5.2. (Global knowledge exchange case). *Let $P^*(t)$ be the probability that the BCO algorithm generates an optimal solution $\boldsymbol{x}^*$ at least once within the first t iterations. Let $p^*(t)$ be the probability that in the t-th iteration any bee generates the optimal solution although it has not yet been produced. Then for an arbitrary $\varepsilon > 0$ and for a sufficiently large t, $p^*(t) \geq p_{min} > 0$ implies $P^*(t) \geq 1 - \varepsilon$. Asymptotically, this yields*

$$\lim_{t \to +\infty} P^*(t) = 1.$$

Proof: ($\Rightarrow$)
By Def. 10 and assuming $\boldsymbol{x}$ represents $\boldsymbol{x}^*$, we get the same as expression 5.2.

$$\begin{aligned} P^*(t) &= 1 - \Pr(\{\boldsymbol{x}^* \text{ is not found in any of } t \text{ iterations}\}) \\ &= 1 - \Pr(B(t)). \end{aligned} \tag{5.4}$$

To obtain expression for $\Pr(B(t))$ we use definition of $p^*(t)$. Consequently, probability $1 - p^*(t)$ designates probability of an event $B(t)|B(t-1)$, that is

$$1 - p^*(t) = \Pr(B(t)|B(t-1)) = \frac{\Pr(B(t) \cap B(t-1))}{\Pr(B(t-1))} = \frac{\Pr(B(t))}{\Pr(B(t-1))}. \tag{5.5}$$

Multiplying above expression for all $t \geq 1$, we obtain:

$$\prod_{i=1}^{t} (1 - p^*(i)) = \prod_{i=1}^{t} \frac{\Pr(B(i))}{\Pr(B(i-1))} = \frac{\Pr(B(t))}{\Pr(B(0))}. \tag{5.6}$$

The event $B(0) = \Omega$, and therefore, $\Pr(B(0)) = 1$. Hence

$$\Pr(B(t)) = \prod_{i=1}^{t} (1 - p^*(i)). \tag{5.7}$$

By substituting expression (5.7) into (5.4), asymptotically we get

$$1 \geq \lim_{t \to +\infty} P^*(t) = 1 - \lim_{t \to +\infty} \prod_{i=1}^{t} (1 - p^*(i)) \geq 1 - \lim_{t \to +\infty} (1 - p_{min})^t = 1.$$

($\Leftarrow$) Suppose that $\boldsymbol{x}^*$ is found by BCO. This implies existence of strictly positive bound $p_{min} > 0$. ∎

Theorems 5.1 and 5.2 guarantee convergence w. pr. 1. of $f(x^{bsf})$ to f^*. Note that for a given optimization problem there may exist more than one optimal solution. In that case, from theoretical point of view, we consider $\mathcal{X}^* \subseteq \mathcal{X}$ and state that the problem is solved to optimality if at least one optimal solution $\boldsymbol{x}^* \in \mathcal{X}^*$ is found.

5.6 Model convergence of BCO

In this section we concentrate our consideration to the analysis of the BCO meta-heuristic in order to prove that both constructive and improvement variants satisfy model-based search properties. Obviously, we need to consider only the variants of BCO that use global knowledge exchange. The meta-heuristic method can have the model convergence property only if it learns from the previously visited solutions [Gut09, Zlo04] and adjusts parameter values according to that knowledge. Moreover, the model-based BCO algorithm should represent a highly structured search procedure which exploits the historical record of performance reflected at each stage of its execution. All these requirements could be fulfilled if the forward pass includes some learning properties. More precisely, the evaluation of possible moves (step (2)(i)(a) in the BCO pseudo-code given in Fig. 4.4, pg. 60) should depend on the moves already explored in previous iterations.

In order to assure the generation of high quality solutions, we should increase the selection probability for the components previously used to obtain good solutions. Therefore, our selection probability involves two components: the problem specific and the learned one.

5.6.1 Preliminary conditions for model convergence

Some asymptotic properties of GHC algorithms were analyzed in [Jac04b]. The authors, seemingly, refine the results of [Pin84, Pin86] and present necessary conditions for the model convergence of GHC algorithm. However, the methodology is described on a more general level such that it become method-independent. In particular, it is used in [Gut02, Gut09] for analyzing ACO convergence properties. Therefore, we use the same methodology in analyzing the parametric properties of the BCO algorithm.

Assuming that $\boldsymbol{x}$ represents an optimal solution $\boldsymbol{x}^*$ of the considered problem and according to Def. 8 and Def. 10 the convergence of $\boldsymbol{x}_t$ in probability to the set $\mathcal{X}^*$ can be expressed as $\Pr(C(t)) \to 1$ as $t \to \infty$.

By the definition of $B(t)$, and [Jac04b] it follows

$$\Pr(B(t)) \to P(B) = \Pr\left(\left\{\bigcap_{t=1}^{+\infty} B(t)\right\}\right) \text{ as } t \to +\infty.$$

We now reproduce the main theorem for the convergence of GHC algorithm from [Jac04b].

Theorem 5.3. *A GHC algorithm converges in probability to $\mathcal{X}^*$ if and only if the following two conditions are satisfied:*
(i) $\sum_{t=1}^{+\infty} r(t) = +\infty$,
(ii) $\Pr(\{C^c(t)|B^c(t-1)\}) \to 0$ *as* $t \to \infty$. ■

The detailed proof can be found in [Jac04b]. For our analysis of BCO only a part, formulated as the following Lemma, is relevant.

Lemma 1. $\Pr(B) = 0 \Leftrightarrow \sum_{t=1}^{+\infty} r(t) = +\infty$.

Proof: ([Gut09]) Since $B = \bigcap_{t=1}^{+\infty} B(t)$ its probability can be calculated as follows:

$$\Pr(B) = \Pr(B(1)) \cdot \Pr(B(2)|B(1)) \cdot \Pr(B(3)|B(2)) \cdots$$

$$= (1 - r(1)) \cdot (1 - r(2)) \cdot (1 - r(3)) \cdots$$

Therefore

$$\begin{aligned} \Pr(B) &= 0 && (5.8)\\ &\Leftrightarrow \prod_{t=1}^{+\infty}(1 - r(t)) = 0 && (5.9)\\ &\Leftrightarrow \sum_{t=1}^{+\infty} \log(1 - r(t)) = -\infty && (5.10)\\ &\Leftrightarrow \sum_{t=1}^{+\infty} r(t) = +\infty. && (5.11) \end{aligned}$$

where the equivalence between (5.9) and (5.10) is obtained applying logarithm to both sides, while the equivalence between (5.10) and (5.11) follows from the fact that for $r(t) \in [0, 1)$ it holds $\log(1 - r(t)) \leq -r(t)$. ∎

Using Lemma 1 and defined convergence conditions for two variants of GBAS algorithms, Gutjahr was able to prove that conditions (i) and (ii) of Theorem 5.3 are satisfied [Gut02]. In [Gut11] the author provides slightly different conditions of the global convergence. Instead of considering an event of visiting an optimal solution, in the first part of the proof he considers probabilities to visit any feasible solutions. This actually means that Lemma 1 holds for any given feasible solution $\boldsymbol{x}$. The above consideration is also applicable to the variant of the BCO algorithm which is defined in such a way to satisfy the model-based search properties formulated in [Zlo04], i.e. the variants of BCO that can fit into the generic framework presented on pg. 89.

In connection with the model convergence, we identify classification of selection schemes regarding the properties of a optimization problem, i.e., when: all components are not included in the solution representation (problem of p-centers) and all are (TSP, VRP, scheduling problem). Thus, both variants of the BCO algorithm, BCOc and BCOi, may be implemented by adjusting to this classification. In particular, the process of building a complete feasible solution should imply different sets of rules, which is why we present theoretical proofs for four different scenarios of the BCO implementation.

5.6.2 Model convergence of BCOc

5.6.2.1 Model convergence of BCOc when all components are not included

We propose the following modification scheme for the selection probability of component i in the iteration t:

$$p_i(t+1) = \begin{cases} 1 - \lambda_t \cdot (1 - p_i(t)) & \text{if } i \in \boldsymbol{x}_t^{bsf}; \\ \lambda_t \cdot p_i(t) & \text{if } i \notin \boldsymbol{x}_t^{bsf}; \\ p_i(0) & \text{if } i \text{ was not choosen.} \end{cases} \tag{5.12}$$

where $p_i(0)$ denotes the initial (problem specific) probability of choosing component i and λ_t represents the time dependent *learning rate*. The idea is to learn from the previous experience how each component influences the quality of generated solution. If component i was a part of the best (best-so-far) solution, the probability that it will be selected in the next iteration is increased. If this component is included in some low quality solutions, we decrease the probability of its selection in the next iteration. Obviously, if component i is not included in any solution, we cannot evaluate its influence. In that case, we keep its selection probability at the initial value. As the number of iterations increases, the selection probability for components not included in the search will become larger than the probability for components belonging to the low quality solutions. Therefore, for these new components, chances to be included in some future solutions will increase. Consequently, as the number of iterations t tends to infinity only the first two modification cases will remain.

Now we can present the sufficient conditions that the BCOc algorithm, with the above defined probability modification scheme, converges in probability to an optimal solution.

Theorem 5.4. *Assume that*

$$1 \geq \lambda_t \geq \frac{\log t}{\log(t+1)} \text{ for all } t \geq t_0, \ (t_0 \geq 2) \tag{5.13}$$

and

$$\sum_{t=1}^{+\infty}(1-\lambda_t) = +\infty. \tag{5.14}$$

Then the corresponding BCOc algorithm converges in probability to one of the optimal solutions from $\mathcal{X}^*$.

Proof: We have to prove that conditions (i) and (ii) from Theorem 5.3 are satisfied.

(i) We will actually prove the equivalent condition (according to Lemma 1), i.e., that $\Pr(B) = 0$. Let us remind that $C(t)$ denotes the event that iteration t is the first in which some solution $\boldsymbol{x}$ (including optimal) is found by some bee. In other words, first we show that for each solution $\boldsymbol{x}$ there exists with probability one an iteration t such that this solution will be generated by any bee. Such property is sufficient to establish best-so-far convergence, since $\boldsymbol{x}^{bsf}$ holds best found solution that is only updated under rigorous improvements.

Consider a fixed solution $\boldsymbol{x}$. Then it holds

$$B = C^c(1) \cap C^c(2) \cap \cdots \Rightarrow \boldsymbol{x} \text{ is never found}$$

and hence

$$\begin{aligned}
\Pr(B) &= \Pr(\{C^c(1) \cap C^c(2) \cap \cdots\}) \leq \Pr(\{\boldsymbol{x} \text{ is never found}\}) \\
&= \prod_{t=1}^{+\infty} \Pr(\{\boldsymbol{x} \text{ is not found in iteration } t \mid \boldsymbol{x} \text{ is not found in iteration } k < t\}) \\
&= \Pr\left(C^c(1)\right) \cdot \prod_{t=2}^{+\infty} \Pr\left(C^c(t) \mid \bigcap_{k=1}^{t-1} C^c(k)\right).
\end{aligned} \tag{5.15}$$

Next, the lower bound for the probability rule of a fixed component is derived. In the worst case, probability to choose component i is decreasing over iterations (becomes lower then the initial probabilities).

Since $\boldsymbol{x}$ can be any feasible solution we use worst case scenario (the one that is decreasing probability of its generation) which is modeled by selecting components that don't belong to $\boldsymbol{x}^{bsf}$ in the probability update rule (5.12). Therefore, for any component i it holds:

$$p_i(t) = \left[\prod_{k=1}^{t-1} \lambda_k\right] \cdot p_i(0),$$

which can be easily verified by induction. Let $t_0 \geq 2$. Then from the condition (5.13) and for $t > t_0$ it holds

$$\begin{aligned}\left[\prod_{k=1}^{t-1}\lambda_k\right]\cdot p_i(0) &\geq \left[\prod_{k=1}^{t_0-1}\lambda_k\right]\cdot\left[\prod_{j=t_0}^{t-1}\frac{\log j}{\log(j+1)}\right]\cdot p_i(0)\\ &= \left[\prod_{k=1}^{t_0-1}\lambda_k\right]\cdot\frac{\log t_0}{\log t}\cdot p_i(0)=\frac{C}{\log t}.\end{aligned}$$

where $C = \left[\prod_{k=1}^{t_0-1}\lambda_k\right]\cdot\log t_0\cdot p_i(0) = const$. The above inequality provides lower bound on the probability $p_i(t)$ that in iteration $t > t_0$ a bee will choose component i. Consecutively, for the probability to find the solution $\boldsymbol{x}$ by any bee it holds:

$$\prod_{i\in\boldsymbol{x}} p_i(t) \geq \left(\frac{const}{\log t}\right)^{\#\boldsymbol{x}},$$

and in the case of optimal solution:

$$\prod_{i\in\boldsymbol{x}^*} p_i(t) \geq \left(\frac{const}{\log t}\right)^{\#\boldsymbol{x}^*},$$

where $\#\boldsymbol{x}$ ($\#\boldsymbol{x}^*$) denotes the number of components constituting the solution $\boldsymbol{x}$ ($\boldsymbol{x}^*$).

Considering the complementary event (the solution $\boldsymbol{x}$ is never found by any bee in $t > t_0$) we obtain the upper bound on the right hand side of the relation (5.15) as:

$$\prod_{t=t_0}^{+\infty}\left[1-\left(\frac{const}{\log t}\right)^{\#\boldsymbol{x}}\right].$$

Applying a logarithm to this expression and using the convexity of the exponential function expressed by $(1-a)\leq e^{-a}, \forall a\in(0,1)$ (i.e., $\log(1-a)\leq -a$) gives us:

$$\sum_{t=t_0}^{+\infty}\log\left(1-\left(\frac{const}{\log t}\right)^{\#\boldsymbol{x}}\right)\leq -\sum_{t=t_0}^{+\infty}\left(\frac{const}{\log t}\right)^{\#\boldsymbol{x}}=-\infty.$$

From this we can conclude

$$\prod_{t=t_0}^{+\infty}\left[1-\left(\frac{const}{\log t}\right)^{\#\boldsymbol{x}}\right]=0,$$

i.e., in (5.15) we have $\Pr(B)\leq 0$. Since $\Pr(B)\geq 0$ always holds, we can conclude $\Pr(B)=0$.

First part of the proof concludes the best-so-far convergence, i.e., with probability one there is an iteration $t > t_0$ where solution x is found. It implies that with the proposed selection rule any feasible solution (optimal included) is reachable. This also coincides with the claim (i) of the Theorem 5.3, i.e., probability that the optimal solution will never be generated tends to zero for large enough t. Second part of the proof concerns model convergence. The goal is to provide proof that after reaching the optimal solution the BCO model will support only the generation of optimal solutions by initiating update rule (5.12). In other words, $p_i(t)$ converges to p_i^* defined as

indicator function:

$$p_i^* = \begin{cases} 1 & \text{if } i \in \boldsymbol{x}^*; \\ 0 & \text{if } i \notin \boldsymbol{x}^*. \end{cases}$$

(ii) To begin our consideration, let us verify that the probability modification rule (5.12) increases the selection probability for components $i \in \boldsymbol{x}^*$ to the maximum. Let m denote the index of the iteration when $\boldsymbol{x}^*$ is generated for the first time. Then in all iterations $t > m$, the selection probability for components included in the optimal solution converge to unity as the number of iterations tends to infinity. Indeed, these probabilities are updated in some iteration $m + r, r = 1, 2, ...$ according to the formula:

$$p_i(t) = 1 - \prod_{k=m+1}^{m+r} \lambda_k \cdot (1 - p_i(m)).$$

This can be easily justified by induction. Due to the condition (5.14) we have

$$\sum_{t=1}^{+\infty}(1 - \lambda_t) = +\infty, \quad \text{which is equivalent to,} \quad \prod_{k=1}^{+\infty} \lambda_k = 0.$$

Therefore, once the optimal solution is found for the probability that the component $i \in \boldsymbol{x}^*$ will be used again it holds:

$$\lim_{t \to +\infty} p_i(t) = 1 - \lim_{t \to +\infty} \left[\prod_{k=m+1}^{t-1} \lambda_k \right] \cdot (1 - p_i(m)) = 1.$$

Consider the component $i \notin \boldsymbol{x}^*$. According to (5.12), its selection probability after iteration m, i.e., in some iteration $m + r$, $r = 1, 2, ...$ is modified as follows:

$$p_i(m + r) = \left[\prod_{k=m+1}^{m+r} \lambda_k \right] \cdot p_i(m).$$

After generating the optimal solution $\boldsymbol{x}^*$, for the probability that the component $i \notin \boldsymbol{x}^*$ will be used again the condition (5.14) yields:

$$\lim_{t \to +\infty} p_i(t) = \lim_{t \to +\infty} \left[\prod_{k=m+1}^{t-1} \lambda_k \right] \cdot p_i(m) = 0,$$

which completes the proof of the theorem. ∎

5.6.2.2 Model convergence of BCOc when all components are included

The above consideration was focused on the convergence properties of the constructive variant of the BCO algorithm in the case where not all the components are included in the solution. Next, we prove model convergence of the BCOc algorithm when all the components are included in the solution. This scenario occurs while dealing with traveling salesman problem, vehicle routing problems or scheduling problems. As in previous section, in order to assure the generation of high quality solutions, we should change the selection probability for the components based on previously obtained good

solutions.

Let us consider TSP and denote by (i,j) a pair of components that are directly connected. We propose the following modification scheme for the selection probability of component j in the iteration t after we have chosen component i:

$$p_{(i,j)}(t+1) = \begin{cases} 1-\lambda_t \cdot (1-p_{(i,j)}(t)) & \text{if } (i,j) \in \boldsymbol{x}_t^{bsf}; \\ \lambda_t \cdot p_{(i,j)}(t) & \text{if } (i,j) \notin \boldsymbol{x}_t^{bsf}; \end{cases} \tag{5.16}$$

where λ_t represents the time dependent learning rate. The idea is to learn from the previous experience how each component influences the quality of generated solution. If the pair of components (i,j) was a part of the current best (best-so-far) solution, the probability that it will be selected in the next iteration is increased. If this pair of components is included in some low quality solutions, we decrease the probability of its selection for the next iteration. Now we can present the sufficient conditions that the BCOc algorithm, with the above defined selection probability modification scheme, converges in probability to an optimal solution.

Theorem 5.5. *Assume that*

$$1 \geq \lambda_t \geq \frac{\log t}{\log(t+1)} \text{ for all } t \geq t_0, \ (t_0 \geq 2), \tag{5.17}$$

and

$$\sum_{t=1}^{+\infty}(1-\lambda_t) = +\infty, \tag{5.18}$$

Additionally, $p_{(i,j)}(0)$ represents initial set of probabilities. Then the corresponding BCO algorithm converges in probability to one of the optimal solutions from $\mathcal{X}^*$.

Proof: The proof is almost identical to the proof of Theorem 5.4. The difference resides within the number of components, which is why we present all the necessary steps. We start by proving that conditions (i) and (ii) from Theorem 5.3 are satisfied.

(i) We prove the equivalent condition (according to Lemma 1), i.e., that $\Pr(B) = 0$. Consider a fixed solution $\boldsymbol{x}$. $C(t)$ denotes the event that iteration t is the first in which a solution $\boldsymbol{x}$ is found by some bee. Then it holds

$$B = C^c(1) \cap C^c(2) \cap \cdots \Rightarrow \boldsymbol{x} \text{ is never found}$$

and hence

$$\begin{aligned} \Pr(B) &= \Pr(\{C^c(1) \cap C^c(2) \cap \cdots\}) \leq \Pr(\{\boldsymbol{x} \text{ is never found}\}) \\ &= \prod_{t=1}^{+\infty} \Pr(\{\boldsymbol{x} \text{ is not found in iteration } t | \boldsymbol{x} \text{ is not found in iteration } k < t\}). \end{aligned} \tag{5.19}$$

Let us now return to the components and assume that the pair (i,j) was not established in iterations $1,\ldots,t$. In the worst case and according to the cumulative probability update rule (5.16), for all pairs of components (i,j) it holds:

$$p_{(i,j)}(t) = \left[\prod_{k=1}^{t-1} \lambda_k\right] \cdot p_{(i,j)}(0),$$

which can be easily verified by induction. From the condition (5.17) it holds

$$\begin{aligned}\left[\prod_{k=1}^{t-1} \lambda_k\right] \cdot p_{(i,j)}(0) &\geq \left[\prod_{k=1}^{t_0-1} \lambda_k\right] \cdot \left[\prod_{j=t_0}^{t-1} \frac{\log j}{\log(j+1)}\right] \cdot p_{(i,j)}(0) \\ &= \left[\prod_{k=1}^{t_0-1} \lambda_k\right] \cdot \frac{\log t_0}{\log t} \cdot p_{(i,j)}(0) = \frac{\text{const}}{\log t}.\end{aligned}$$

Therefore, the above derived is a lower bound of the worst case selection scenario for any component (i,j). Consecutively, for the probability to find the solution $\boldsymbol{x}$ by any bee it holds:

$$\prod_{(i,j)\in\boldsymbol{x}} p_{(i,j)}(t) \geq \left(\frac{const}{\log t}\right)^n,$$

where n denotes the total number of components (number of cities in the TSP). Considering the complementary event (the solution $\boldsymbol{x}$ was not found by any bee) we obtain the upper bound on the right hand side of the relation (5.19) as:

$$\prod_{t=t_0}^{+\infty} \left[1 - \left(\frac{const}{\log t}\right)^n\right].$$

Applying a logarithm to this expression gives us:

$$\sum_{t=t_0}^{+\infty} \log\left(1 - \left(\frac{const}{\log t}\right)^n\right) \leq -\sum_{t=t_0}^{+\infty} \left(\frac{const}{\log t}\right)^n = -\infty.$$

From this we can conclude

$$\prod_{t=t_0}^{+\infty} \left[1 - \left(\frac{const}{\log t}\right)^n\right] = 0,$$

i.e., in (5.19) we have $\Pr(B) \leq 0$. Since $\Pr(B) \geq 0$ always holds, we can conclude $\Pr(B) = 0$.

(ii) (Model convergence) This condition means that if m is the index of the iteration when $\boldsymbol{x}^*$ is generated for the first time, then $\boldsymbol{x}_t^{bsf} = \boldsymbol{x}^*$ for all $t \geq m$. Moreover, in all iterations $t > m$ the selection probability for pairs of components, not included in the optimal solution, converge to zero as the number of iterations tends to infinity.

Consider the pair of components $(i,j) \notin \boldsymbol{x}^*$. According to (5.16), its selection probability after iteration m, i.e., in some iteration $m+r$, $r = 1, 2, \ldots$ is modified as follows:

$$p_{(i,j)}(m+r) = \left[\prod_{k=m+1}^{m+r} \lambda_k\right] \cdot p_{(i,j)}(m).$$

Due to the condition (5.18) we have

$$\sum_{t=1}^{+\infty}(1-\lambda_t)=+\infty, \quad \text{which is equivalent to,} \quad \prod_{k=1}^{+\infty}\lambda_k=0.$$

Therefore, after generating the optimal solution $\boldsymbol{x}^*$, for the probability that the pair of components $(i,j)\notin\boldsymbol{x}^*$ will be again used it holds:

$$\lim_{t\to+\infty}p_{(i,j)}(t)=\lim_{t\to+\infty}\left[\prod_{k=m+1}^{t-1}\lambda_k\right]\cdot p_{(i,j)}(m)=0,$$

which completes the proof of the theorem. ∎

Considering the problem of scheduling independent tasks on identical processors, where the pair of components (i,j) describes the situation that task i is allocated to processor j, we can apply the selection probability modification scheme defined by (5.16). Therefore, the above described reasoning holds, and the resulting BCO algorithm satisfies model convergence properties.

5.6.3 Model convergence of BCOi

The above consideration was mainly related to the BCOc algorithm. However, it can be easily generalized to BCOi, as we distinguish two cases w.r.t. number of components constituting a final solution. Main difference between the two modification schemes is the ordering. For the problems like TSP, where feasible solutions contain all components, the probability update rules should be modify in such a way to consider ordering of components. In the other case, modifications of solutions only need to consider components' affiliation within the x^{bsf}.

We present model convergence for BCO for the first time. There are no previous publications of presented material. We first propose modification scheme (5.20) inspired by the update rules, similar to the one provided for BCOc when all components are not included in the complete solutions. The cumulative probability update rule depends on the components that will be taken out of the current solution, as well as on the components identified for the inclusion in the current solution. It is presumed that the best-so-far reinforcement is used as provided in Formula (5.1).

The following modification scheme for the selection probability that component i should replace component j in the iteration t is as follows:

$$p_{(i,j)}(t+1)=\begin{cases}1-\lambda_t\cdot(1-p_{(i,j)}(t)) & \text{if } i\in\boldsymbol{x}^{bsf} \text{ and } j\notin\boldsymbol{x}^{bsf};\\ \lambda_t\cdot p_{(i,j)}(t) & \text{otherwise.}\end{cases} \tag{5.20}$$

Next, let us consider the case of optimization problems where all components of the problem constitute a solution. We start with problems such as TSP, where an ordering of solution's components needs to be accounted. Moreover, let there be an ordering such that city j is not behind city i. The probability to move city j behind i should be determined as follows:

$$p_{(i,j)}(t+1)=\begin{cases}1-\lambda_t\cdot(1-p_{(i,j)}(t)) & \text{if } (i,j)\in\boldsymbol{x}^{bsf};\\ \lambda_t\cdot p_{(i,j)}(t) & \text{otherwise.}\end{cases} \tag{5.21}$$

Furthermore, when considering problems such as scheduling or VRP, we can denote the pair (i, j) as indicator that a task i should be allocated on the processor j, i.e., location i is associated to vehicle j. Therefore, the modification scheme (5.21) can be applied for this kind of problems.

Finally, the theorem identifying the conditions under which the BCOi algorithm, with the above defined selection probability modification schemes (5.20) and (5.21), converges in probability towards optimal solution is identical for all considered cases of optimization problems.

Theorem 5.6. *Assume that*

$$1 \geq \lambda_t \geq \frac{\log t}{\log(t+1)} \text{ for all } t \geq t_0, \ (t_0 \geq 2), \tag{5.22}$$

and

$$\sum_{t=1}^{+\infty}(1-\lambda_t) = +\infty, \tag{5.23}$$

Additionally, $p_{(i,j)}(0)$ represents initial set of probabilities. Then the corresponding BCOi algorithm converges in probability to one of the optimal solutions from $\mathcal{X}^*$.

Proof: The proof is identical to the proof of Theorem 5.5. ∎

In addition, let us elaborate the existence of the proposed probability modification scheme. For example, if learning rate $\lambda_t = 1 - c/(t \log t)$ for a given constant $0 < c < 1$, the conditions of Theorems 5.4–5.6 are valid.

5.7 Final remarks

Based on the existing methodology, we distinguish two types of convergence, a weaker best-so-far convergence, and a stronger model convergence. In this chapter we prove best-so-far convergence of general BCO algorithm for arbitrary optimization problem by providing sufficient and necessary conditions that satisfy the requirement $p^* > 0$. In both cases of BCO, with independent as well as dependent iterations, we conclude that it is enough if the BCO algorithm assures that any feasible solution may be generated during the search process. However, the assumption is weak in order to reason about the consistency of the BCO algorithm, which is: after providing an optimal solution the algorithm needs to support production of this optimal solution in the next iterations. Therefore, we establish the conditions that are sufficient for the model convergence of the BCOc and BCOi by following the guidelines presented in section 5.3.2. The conditions of the model convergence are: 1. convergence in the best-so-far sense, i.e., all feasible solutions are reachable from any position in the search space; 2. the generation of an optimal solution is favored once it is found. We have fulfilled the requirements by providing rules of construction/modification that exploit historical record of the BCO algorithm performance. We provide four modification schemes w.r.t. type of considered optimization problem and variant of the BCO algorithm. In order to analyze the algorithm's efficiency, the next step is to examine the speed of the model convergence, i.e., to investigate the estimation of $E(T_1)$, the expected time of the first hit for an optimal solution (pg. 86).

5.8 Chapter summary

The chapter was devoted to the topic of theoretical analysis of the BCO meta-heuristic method. We begin by providing a survey of the most recent developments in this field of optimization that helps to establish conditions under which a meta-heuristic converges towards an optimal solution.

BCO has been extensively developed and explored in the last decade [Dav11a, Dav12, Eda08, Mar07, Nik13a, Teo05, Teo08, Teo06, Teo07, Šel10, Won10b] and has numerous successful applications. However, the theoretical verification (convergence analysis) of BCO was missing and therefore, our aim is to contribute to this topic. The main goal of this dissertation is to provide mathematical verification of the BCO meta-heuristic by examining the conditions of its convergence. We analyze both types of convergence, best-so-far and model-convergence, and provide the hints for constructing the BCO method that converges to optimal solutions.

In this chapter we analyze convergence properties of BCO by providing theoretical proof of asymptotic convergence for two variants of the BCO algorithm: BCOc and BCOi. The topics are arranged in the following manner:

- We begin with the description of BCO within the generic framework [Gut09] and introduce notation as an extension of the material introduced in Chapter 5 or incorporated from the literature.
- We prove that the BCO method is well founded and converges in the best-so-far sense.
- We prove model convergence of BCO. We conclude that the conditions, under which BCO will converge, need to uphold the probabilistic rules that assure algorithm's convergence in the best-so-far sense and that the generation of the optimal solution is favored once it is found. We show that the conditions are fulfilled if the BCO algorithm exploits its historical record of performance by incorporating modification scheme presented in this chapter.

CHAPTER 6

Parallelization strategies for the BCO algorithm

Parallel computing implies simultaneous execution of several tasks (processes) on different processors with the goal to solve faster a given problem instance. Parallelization signifies decomposition of total computational work and the distribution of corresponding tasks (workload) among free processors. The decomposition can be realized at the task level, data level, instruction level or bit level [Bar15]. One of the challenges of parallel computation are balancing the workload and minimizing the communication. It should be noted that in recent literature an expression *concurrent computing* has been used, but should not be confused with parallel computing. Two tasks are concurrent if they can run in overlapping time periods. Readers are referred to [Alb05, Ray12] on the topic of concurrent computing.

The BCO method appears to be suitable for parallelization as it operates on a population of solutions. In this chapter we study this proposition and investigate different parallelization strategies for the BCO method. We address the questions about properties relevant for efficient implementation on distributed memory systems. The chapter begins pointing out the main goals of the parallelization. Next, we review parallelization strategies proposed for meta-heuristic methods. We enclose the survey with description of parallel implementation of two swarm intelligence algorithms, ABC and ACO. Section 6.3 is devoted to description and implementation of parallelization strategies for the BCOc algorithm. In Section 6.4 we provide results of the comparison study between five parallel implementation of BCOc for $P||C_{max}$.

6.1 Motivation for parallelization of meta-heuristics

The advancement of exact solution methods has led to a more efficient generation of optimal solutions. However, for a large number of situations that arise in practice there are still many challenges one has to deal with because optimal (or high quality) solutions can be hard to obtain in a reasonable amount of time. As a result, solving optimization problems requires constant development of fast solution methods. To construct faster algorithmic solutions is to use parallel implementations of (existing) sequential algorithms, for shared-memory or distributed-memory systems. An easier access to commodity and high-performance computers has greatly facilitated the development of parallel implementations that help enhance efficiency of generating high

quality solutions.

Majority of well-known meta-heuristic methods have been parallelized for various optimization problems [Alb05]. This greatly contributed to the standardization within the new field of computation that deals with parallelization strategies for meta-heuristics. The population-based meta-heuristics seem to be the most suitable for parallelization. The nature of these methods is to work with multitude of solutions, either by utilizing greedy or a local search method. In the both cases, it is straightforward to make an assumption of an independent workload during the constructions or modifications of solutions, that could be executed on different processors. This scenario corresponds to one of the ways we can implement a parallelization strategy.

The main goal of parallelization is to speedup the computations needed to solve a particular problem by engaging several processors and dividing the total amount of work between them. For stochastic algorithms (meta-heuristics, in particular), several goals may be achieved, such as [Alb06, Tal09, Cra14]:

- speeding up the search (i.e., reducing the search time);
- improving the quality of the obtained solutions (by enabling searching of different parts of the solution space);
- improving the robustness (in terms of solving different optimization problems and different instances of a given problem, in an effective manner, robustness may also be measured in terms of the sensitivity of the meta-heuristic to its parameters);
- solving large-scale problems (i.e., solving very large instances that cannot be solved by a sequential machine).

A combination of gains may also be obtained: parallel execution can enable an efficient search through different regions of the solution space, yielding an improvement of the quality of the final solution within a smaller amount of execution time.

A significant amount of work concerning the parallelization of meta-heuristics already exists. The approach can be twofold, considering theoretical aspects of parallelization [Ste02], or developing practical applications of parallel meta-heuristics for different optimization problems. The survey papers [Ver95, Cun02, Cra05, Cra07, Cra10, Cra14] summarize these works and propose an adequate taxonomy.

6.2 Parallelization strategies of meta-heuristics

According to [Cra05], parallelization strategies can be conducted in different ways. The first classification of parallelization strategies for meta-heuristic methods is given in [Ver95]. Based on the control of the search process, this classification resulted in two main groups of parallelization strategies: *single walk* and *multiple walks parallelism*. To refine the classification of parallelization strategies, besides the control of the search process, one has to consider communication aspects (synchronous or asynchronous) and search parameters (same or different initial point and/or same or different search strategies). The resulting classification is described in details in [Cra05] and is briefly recalled here in order to enable adequate classification of parallelization strategies for ABC, ACO and BCO.

The classification from [Cra05, Cra07] takes into account three main aspects of parallel execution: search control, communication control, and search differentiation. Such an approach resulted in the 3D-Taxonomy $\mathcal{X}/\mathcal{Y}/\mathcal{Z}$. Here, $\mathcal{X}$ is used to denote *search control cardinality*, which could take centralized ($1C$) or distributed (pC) values. $\mathcal{Y}$ deals with two aspects of *communication control*, synchronization and type of data to be exchanged. The four possibilities for $\mathcal{Y}$ are Rigid Synchronous (RS), Knowledge Synchronous (KS), Collegial Asynchronous (C) and Knowledge Collegial (KC). The RS classification is customary for independent executions and synchronous strategies where there is no need to apply local changes to the distributed information. The KS classification depicts strategies where a certain knowledge is utilized on the distributed data [Cra05, Cra14]. Collegial classification refers to asynchronous strategies with the same properties described previously. *Search differentiation* $\mathcal{Z}$ specifies the part of the search executed by each of the parallel processes. The difference is characterized by the initial point and by the search strategy. Each process can start from the same or different initial point, and it can perform the same or different search procedure. Therefore, there exist four combinations for $\mathcal{Z}$: Same initial Point-Same search Strategy (SPSS), Same initial Point-Different search Strategies (SPDS), Multiple initial Points-Same search Strategy (MPSS), Multiple initial Points-Different search Strategies (MPDS). The particular implementation of each of the described strategies may vary, depending on the given multiprocessor architecture and the characteristics of the problem at hand.

BCO is closely related to some other swarm intelligence meta-heuristic methods, especially ABC [Kar05, Kar07] and ACO [Dor99, Dor10]. ABC and ACO are more extensively exploited in the recent literature, in both sequential and parallel variants. Therefore, we review the papers dealing with the parallelization of these two meta-heuristics and exploit their methodologies here. However, some new strategies are also proposed.

6.2.1 Parallelization of ABC

There are several papers in the recent literature describing parallelization techniques for ABC. A parallel version of the ABC algorithm for shared memory architectures was presented in [Nar09]. It was shown that the proposed parallelization strategy did not degrade the quality of obtained solutions, and achieved substantial speedup. There, the entire colony of bees was divided equally among the available processors. A set of solutions was placed in the local memory of each processor, while a copy of each solution was maintained in the global shared memory. During each cycle the set of bees associated with each processor, improved the solutions in its local memory. At the end of the cycle, the solutions were copied into the corresponding slots in the shared memory and made available to all other bees. A similar approach was proposed in [Ban10], and was implemented on a distributed memory multiprocessor system. The authors decomposed the entire bee colony into several subgroups, and each subgroup performed a local search concurrently on each processing node. The local best solutions were exchanged between the nodes. The algorithm implementation utilized the message passing technique as the communication paradigm. The experimental results showed improvement in both solution quality and computing time, when compared to the sequential ABC algorithm. We would classify both aforementioned strategies as

pC/RS/MPDS.

The authors of [Par11] presented three parallel models for the ABC algorithm: a master-slave approach that divides the processing load into several processors; a multi-hive approach that promotes periodic migrations between independent sub-populations; and a hierarchical approach that hybridizes the two former models. The MPI communication library, different variants of the ABC algorithm, and a large number of bees (over 1000) were used. The Master-Slave ABC (MS-ABC) model was a global single-population system where a master process divided the computational effort into several slave processes, each one running on a different processor. After completing their assigned jobs, slaves would send the results to the master process. According to the adopted classification, this model falls into the 1C/RS/SPSS category. A Parallel Multi-Hive model (MH–ABC) was a multiple-population coarse-grained system that used two or more hives initialized at the same time, with different random seeds. Therefore, this model belongs to the pC/RS/MPDS category. The combination of these two methods was also given as the Parallel Hybrid Hierarchical Model (HH–ABC), which had two levels: multiple-population coarse-grained islands were executed at the upper level, while single-population master-slaves were invoked at the lower level. The authors showed that MH–ABC was able to outperform the sequential ABC algorithm.

Three different approaches to the parallelization of ABC were proposed in [Sub11]: parallel independent runs; multiple swarms – one best solution; and multiple swarms – best solutions from all swarms. Increasing performance was the main focus of the parallel independent runs approach, while the multiple swarm approaches aimed at obtaining better results. In the parallel independent runs approach, every thread ran the same sequential ABC algorithm, with different random seeds. The final solution was the best one from all the independent runs. The speedup was almost linear. This strategy can be classified as pC/RS/SPDS. The other two approaches were based on multiple swarm tactics, and the idea was to use more than one swarm on the same search space. These swarms were able to communicate with each other in order to exchange their best-so-far solutions. The number of cycles between two communications was determined as the ratio between the total number of cycles and the number of swarms. In the "multiple swarms – one best solution" approach only the best among all "best-so-far solutions" was accepted by all other swarms, and therefore, we classify this approach as pC/KS/SPDS. In the "multiple swarms – best solutions from all swarms" approach "best-so-far solutions" from all swarms were exchanged and influence the further execution of the parallel ABC algorithm. Therefore, this approach should be classified as pC/KS/MPDS. On a set of eleven well-known benchmark functions, the authors reported significant speedup of the independent run, as well as an improvement in the quality and consistency of the final solutions obtained by multiple swarm parallelization strategies.

6.2.2 Parallelization of ACO

Parallel ACO algorithms exist in the literature for more than twenty years, starting with a very fine-grained parallelization on the Connection Machine proposed by [Bol93]. Two parallelization strategies for ACO applied to the Travelling Salesman Problem (TSP) were proposed in [Stü98]. The first involved independent executions of a single ACO with different seeds, classified as pC/RS/MPSS. The second strategy aimed

at speeding up the local search procedure, computationally the most intensive part of the ACO algorithm. It falls into medium grained parallelization strategies and can be classified as 1C/KS/MPSS. The reported experimental results showed that for the independent execution almost linear speedup could be obtained, if a small number of processors was used (up to 6). The second strategy did not prove efficient for TSP due to the fact that sequential local search was fast enough. Two parallel implementations of ACO for Set Covering Problem were proposed in [Rah02]. Independent execution on up to 40 processors showed the improvement of the solutions found by the sequential algorithm. In addition, the average speedup 37.17 and the average efficiency 92.93% was obtained. The second implementation involved a very fine-grained parallelization (1C/KS/MPSS) with each ant considered as a separate search process. The average speedup of 10.95 was obtained on 20 processors. In addition to the independent execution, various coarse grained cooperative parallel implementations of ACO for TSP were proposed in [Man06]. The authors tested both synchronous (pC/KS/MPSS) and asynchronous (pC/C/MPSS) communication concepts realized on four interconnection topologies: fully connected, replace worst, hypercube, and ring. Artificial colonies from different processors exchanged only the best-so-far solution. The presented experimental evaluation indicated that independent execution was the best performing approach under the tested conditions. Similar approach was used in [Jov09] to tackle Minimum Weighted Vertex Cover Problem. Four topologies (fully connected, replace worst, and two rings) executing synchronous multi colony ACO approach were tested and compared against the independent and sequential executions. Contrary to [Man06], the authors of [Jov09] concluded that the ring topology performed the best.

A cooperative multi colony approach applied to TSP and Quadratic Assignment problem was proposed in [Mid02]. Synchronous communication involved one or several best solutions and therefore this approach can be classified as pC/KS/MPSS. The authors investigated different ways of exchanging solutions among ant colonies. They considered an exchange of the global best solutions among all colonies and local exchanges based on a virtual neighborhood between two colonies which corresponds to a directed ring. Their main observation was that the best results, with respect to computing time and solution quality, were obtained by limiting the information exchange to a local neighborhood of two colonies.

The synchronous multi colony ACO approach using an unidirectional ring topology of 8 processors was used in [Yan07c, Yu11]. Each colony performed an independent search for a given number of iterations called an epoch. Then, a specified number of high quality solutions were propagated through the processor ring and used for the pheromone trial updates. The described approach can be classified as pC/KS/MPSS. The reported experimental evaluation showed that both the solution quality and the execution time were improved in the parallel ACO when compared to a single colony model.

For a more extensive survey of parallel ACO implementation, the readers are referred to [Dor10, Ped11].

6.3 Parallelization strategies of the BCO algorithm

The BCO algorithm is created as a multi-agent system which inherently provides a good basis for parallelization on different levels. High-level parallelization assumes a coarse granulation of tasks, and can be applied to the iterations of BCO. Smaller parts of the BCO algorithm (the forward and backward passes within a single iteration) occur on the agent level. They are suitable for low-level parallelization because they contain a lot of independent executions. In this dissertation, we consider a coarse granulation strategy, since we are using distributed memory multiprocessor systems. Fine-grained parallelization is not suitable for these multiprocessor systems, as it was verified in [Dav11b]. On shared-memory multiprocessor systems, however, our preliminary experiments showed that we can benefit from the fine-grained strategy by employing OpenMP extension [Dag98]. Usefulness of the OpenMP directives rises from the careful implementation of the synchronization points and utilization of private data structures. If we do not take care of the CPU time needed for creating threads and their synchronization, the conclusion is that fine-grained parallelization does not provide any benefits, as stated in [Sub11].

The first approach to parallelization of BCOc was proposed in [Dav11b]. The authors considered coarse-grained parallelization obtained by dividing the total amount of computations in a single BCOc instance[(1)] among the available processors. We propose three different strategies for parallelization of BCOc, one independent and two cooperative. Independent strategy is generalization of the approach from [Dav11b], as it involves an independent execution of various BCO instances, for which the total amount of computation is (equally) distributed among available processors. The cooperative strategies involve knowledge exchange between various BCOc instances executed on different processors. The main difference between them is in the way the communication is performed (synchronous or asynchronous). Asynchronous strategy of BCOc is the most general parallelization concept as it provides more diversified search. The asynchronous strategy is implemented in two ways, with *centralized* and *non-centralized* information exchange. More details about all the implementations are provided in the remainder of this section.

6.3.1 Independent execution of the BCO algorithms

The simplest form of coarse-grained parallelization of BCO represents the independent executions of necessary computations on different processors, as it is illustrated in Fig. 6.1. More precisely, we distribute all the calculations among the available processors denoted by BCO$_i, i = 1, \ldots, q$, where q is total number of engaged processors. This distribution could be obtained e.g., by the division of the stopping criterion (SC) among the processors. For example, if the stopping criterion is maximum allowed CPU time (given as a $runtime$ value in seconds), we could run BCO in parallel on q processors for $runtime/q$ seconds. A similar rule could be introduced in the case when the stopping criterion is the allowed number of iterations. In both cases, each processor independently performs a sequential variant of BCO, with a different seed, same configuration of the BCO parameters ($B_i = B$, $NC_i = NC, i = 1, \ldots, q$) and with a reduced value of the stopping criterion. The main aim of this approach is to speed up

[(1)] An instance of BCO is obtained by specifying all parameters' values.

the execution of BCO by dividing the total workload among several processors, and therefore it could be classified as pC/RS/MPSS. In [Dav11b], this variant of distributed BCO is applied to BCOc for $P||C_{max}$ problem and named DBCO. DBCO is similar to the second approach proposed in [Par11] for the ABC algorithm.

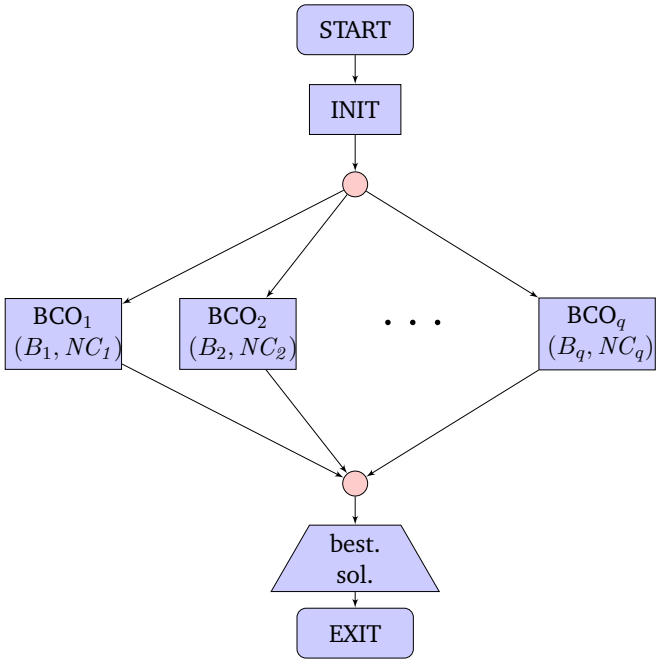


Figure 6.1: Independent execution of several BCO algorithms.

Another way to implement the independent execution of the BCO algorithms could involve dividing the population of bees instead of the stopping criterion. On each processor a sequential BCO is executed with a different seed, equal number of forward/backward passes and with a smaller number of bees. Therefore, if the sequential execution uses B bees for the search, our parallel variant, running on q processors, is using only $B_i = B/q$. This approach was also considered in [Dav11b] and it was referred to as BBCO. BBCO is similar to the first approach proposed for ABC in [Sub11]. In [Dav11b] it is assumed that the BCOc parameters (the number of bees B and the number of forward/backward passes NC) are the same for all BCOc processes executing on different processors, in order to assure load balancing between all processors. Therefore, we classified the BBCO approach also as pC/RS/MPSS.

Combining these two approaches, we can vary the values of the BCO parameters and change the stopping criterion at the same time. We refer to this approach as MBCO (Multiple BCO), classify it as pC/RS/MPDS, and expect it to introduce more diversification into the search process (Fig. 6.1). The changes required to realize distributed execution should be adjusted in such a way as to balance the resulting workload. That could be efficiently implemented if the allowed CPU time is adopted as the stopping criterion, and is reduced q times for parallel execution on q processors.

In such a way, we obtain three variants of independent parallel execution for our BCO, but all of them fit in the block-diagram given in Fig. 6.1. The initialization phase (denoted by INIT on this block-diagram) involves the reading of input parameters and the problem instance, as well as the recalculation of the parameter values according

to the rules defined by the corresponding parallelization variant. After the initialization is completed, each process performs an independent sequential variant of BCO, in the sense that processor k is executing BCO$_k$ ($k = 1, 2, \ldots, q$), which is illustrated by the branching part of the block-diagram. At the end, the best obtained solutions are collected from all processes (processors), and the best among them is reported to the user.

6.3.2 Synchronous cooperation of the BCO algorithms

The more sophisticated way to achieve coarse-grained parallelization is through cooperative execution of several BCO processes. At certain predefined execution points, all processes exchange the relevant data (usually current best solutions). These data are used to guide further searches. This approach, named CBCO, is synchronous and can be classified as pC/KS/MPSS if all BCO processes have the same values of the parameters B and NC. Otherwise, it will belong to the pC/KS/MPDS class. The block-diagram illustrating this strategy would be similar to the one presented in Fig. 6.1, where the branching part should be concatenated several times (depending on the allowed communication frequency). Similar approaches were used for the parallelization of the ABC meta-heuristic [Ban10, Nar09, Sub11].

In order to examine the benefit of the information exchange during the cooperative execution, we do not make a reduction of the stopping criterion, and thus allow all processors to work until the original stopping criterion is satisfied ($SC = const$). The cooperative execution of different BCO algorithms should increase the quality of the search. Therefore, if we do not reduce the allowed CPU time, we expect to obtain an improvement of the quality of the final solution. Of course, the total running time is then increased q times and we have to ensure fair comparison between the sequential and cooperative BCO. We elaborate on this point in the experimental evaluation section.

The communication points can be determined in two different ways [Dav13]: fixed and processor-dependent. The fixed communication means that the best solution is exchanged n_{COM} times during the parallel BCO execution, regardless of the number of processors engaged. In such a way, the processors are given more freedom to perform the associated part of the search. Actually, increasing the number of engaged processors (q) yields an increase in the total number of iterations performed before the communication is initiated. For example, let the stopping criterion be maximal number of iterations, fixed to 1000 (meaning that each processor will perform 1000 iterations and in total $q \cdot 1000$ iterations will be performed). In addition, let $n_{COM} = 10$, i.e., communications are performed when each processor completes 100 iterations. This means that 200 iterations will be performed before the communication is initiated, in the case when $q = 2$. In the case when $q = 5$, the communication occurs after 500 iterations.

The processor-dependent communication frequency provides that the information about the improvement of the current global best solution is passed to all processors more often. For the definition of communication points in this case we used the following rule [Dav13]: the current global best solution is exchanged each $n_{it}/(10 \cdot q)$ iterations where n_{it} represents the maximum allowed number of iterations. This means that in the case when $n_{it} = 1000$ and $q = 5$, the communication will be initiated 50 times during the search, i.e., each time all processors complete $5 \cdot 20 = 100$ iterations. In the

case when $q = 10$, we will have 100 communications, again after $10 \cdot 10 = 100$ iterations in total are completed. Since the stopping criterion is not reduced, each processor will always perform n_{it} iterations, which yields $q \cdot 1000$ in total. In the case when the values of BCO parameters vary from processor to processor, the communication points should be defined with respect to the allowed CPU time, rather than using the number of iterations as the reference value. Increasing the communication frequency may increase the time required for communication and synchronization between processors, which could degrade the performance of the parallel search. The solution of this problem is certainly the asynchronous parallelization of BCO, which we describe next.

6.3.3 Asynchronous cooperation of the BCO algorithms

To decrease the communication and synchronization overhead during the cooperative execution of different BCO algorithms, a more general (GBCO) approach, the asynchronous execution strategy, is proposed [Dav13]. We implement this strategy in two different ways. The first implementation involves a centrally coordinated knowledge exchange, while the second one utilizes non-centralized parallelism. Each processor executes a particular sequential variant of BCO until some predefined communication condition is satisfied. It than informs others about its search status, collects the current global best, and continues its execution. However, this strategy does not require all of the processors to participate in the communication at the same time. Each processor sends its results, and looks for the ones from the others when it is ready.

The first approach assumes the existence of a *central blackboard* (a kind of global memory) [Cra10] to which each processor has access to. The communication condition is defined as *the improvement of the current best solution or the execution of 5 iterations without improvement*. More precisely, each processor performs sequential BCO steps until one of the aforementioned conditions is satisfied. If it manages to improve the current best solution, it informs other processors of that improvement (putting the information on the blackboard) and continues its own execution.

On the other hand, if no improvement occurs after 5 iterations, the corresponding processor checks if there are improvements generated by the other processors. If some other processor reported an improved solution, that new solution is used as the reference point for further search. In the case when an improvement has not been announced by others, the execution continues with the previous best solution as the reference point. The improvement checking frequency (the number of iterations between two consecutive blackboard accesses) is selected in such a way to ensure both individual search on each processor and high information exchange rate.

The stopping criterion is not reduced, and it is set to maximum allowed CPU time, in order to ensure better load balancing, i.e., that all processors complete their execution at approximately the same time. This strategy is classified as pC/C/MPSS in the case when the BCO parameters are the same on all processors (only the seed differs), or as pC/C/MPDS otherwise.

Non-centralized asynchronous parallel BCO execution assumes the existence of several blackboards, and that only a subset of (adjacent) processors may post and access information on the corresponding blackboard. In this case, we allow each processor to perform a single iteration of the corresponding BCO before addressing its associated blackboard. In the case that it manages to improve the current global best solution,

it posts that information on the blackboard and checks if there are better solutions already posted there. The best posted solution is adopted as the new reference point. If the improvement did not occur in the current iteration, the corresponding processor simply checks for a better solution on its associated blackboard. If there is a better solution posted, it serves as a new reference point, otherwise, the execution continues with the previous best as the reference point. This strategy is also classified as pC/C/MPSS or pC/C/MPDS, depending on the search parameters. To the best of our knowledge, asynchronous strategies for bee-inspired were not considered in the recent literature, and therefore, they represent the major contribution in this dissertation.

In Fig. 6.2 we summarize all the strategies and their implementations and provide details that distinguish them. Since some implementations have been already proposed in [Dav11b], here we develop several new parallel implementations (shaded in Fig. 6.2). More details about their experimental evaluation are presented in the next section.

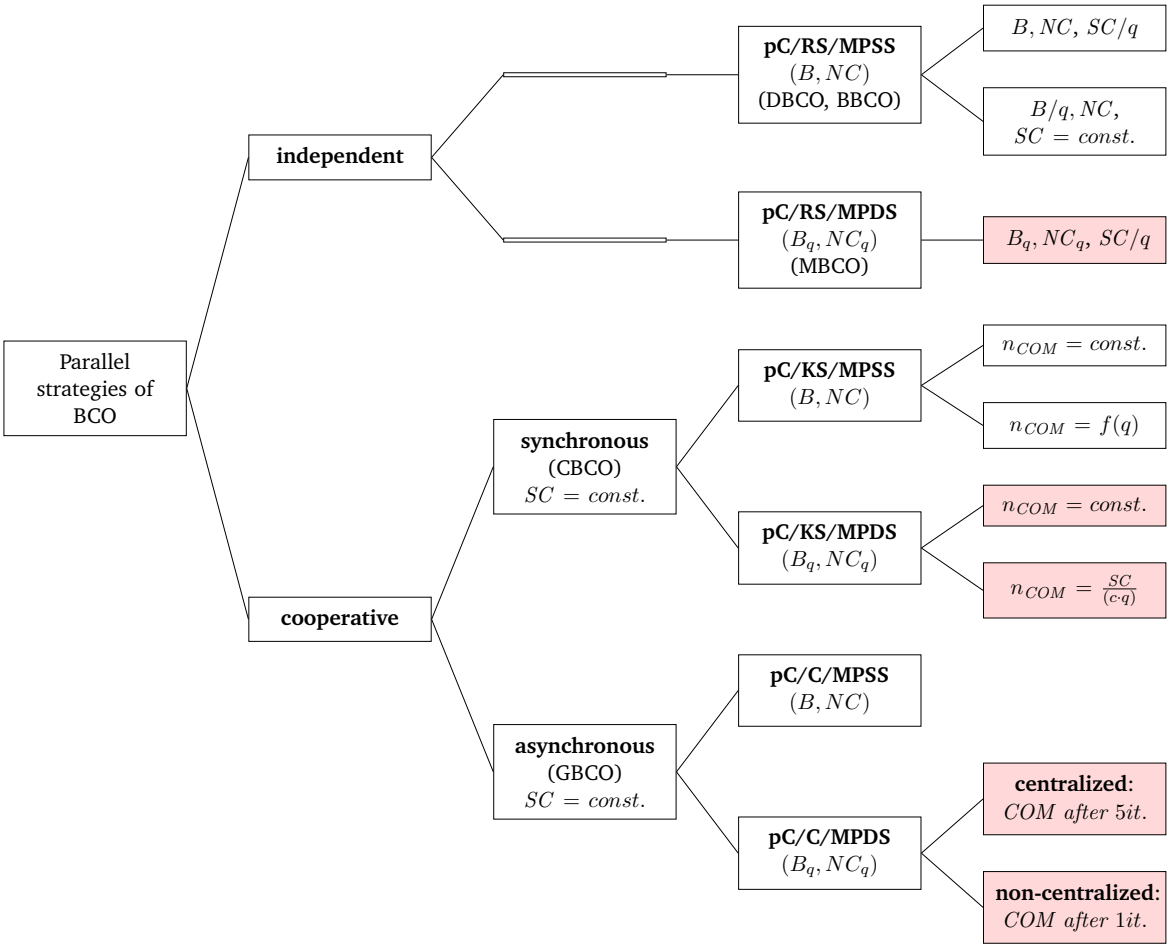


Figure 6.2: Classification of parallelization strategies for BCO. The colored fields indicate settings of our implementations.

6.4 Comparison of the results for parallel BCOc executions

In order to set up the framework for the experimental validation of our approaches to the parallelization of the BCO algorithm, we first describe how the available hardware and software resources influence our implementation. In the rest of this section, we present experimental evaluation of the proposed parallel BCOc implementations for the $P||C_{max}$ problem. The test problem has been introduced in section 2.3.2 and the corresponding sequential BCOc implementation in [Dav12], which is the starting point for the parallelization. Finally, we present the results of a comparison between the sequential and each of the proposed parallel BCO implementations. To ensure fairness of the obtained results, we always compare parallel execution with the best sequential one, executed on a single processor of our parallel architecture (instead of a parallel version executed for $q = 1$).

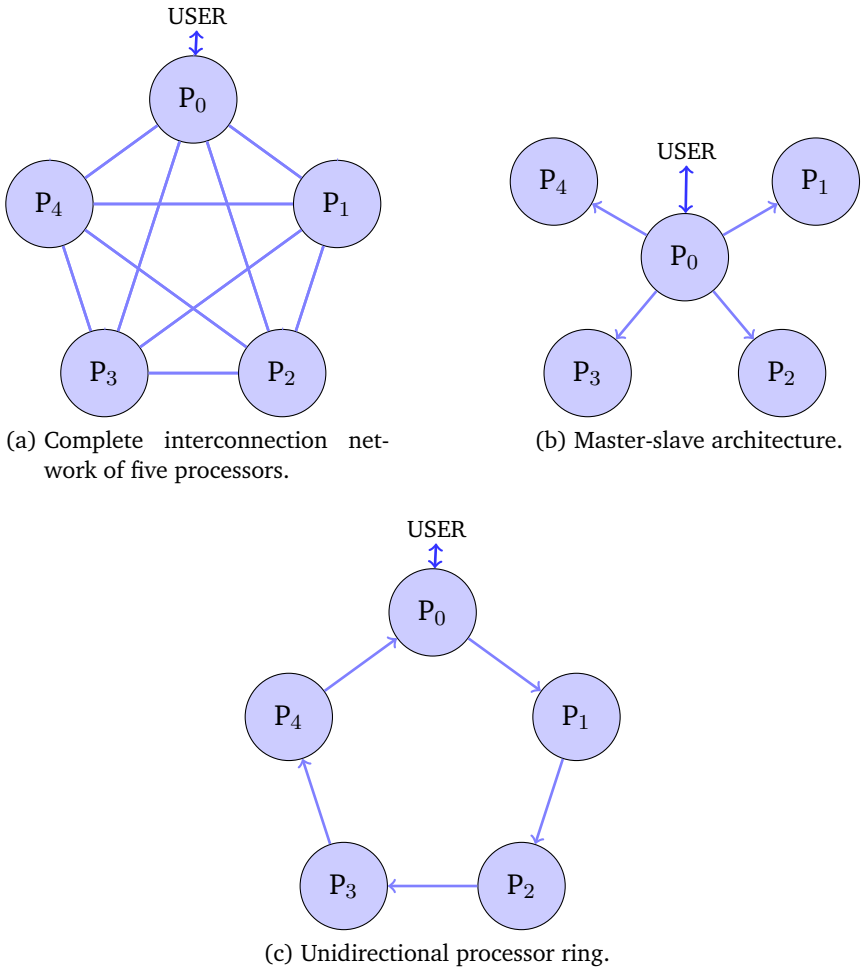
As we already mentioned, BCOc is the stochastic meta-heuristic method, which means that different executions can produce different results, even in the sequential case. Therefore, we cannot use the standard performance measures (speedup and efficiency) for the evaluation of its parallel execution. Moreover, the parallelization changes the original algorithm, and consequently, we need to evaluate both the *execution time* and the *quality of the final solution*. Those become the new performance measures.

6.4.1 Experimental environment

All parallelization strategies are implemented on a distributed memory IBM HPC Linux Cluster: IBM eServer IBM x3650 2 x Dual Xeon 5140 2.0GHz, 4GB RAM, 196GB, and 16 Working Nodes – 2 x Dual Core Intel Xeon 5141 2.33GHz with 4GB memory and 36GB scratch space. Our implementations are coded in the C programming language, using the MPI communication library.

Our target architecture for parallelized BCOc is a homogeneous distributed memory multiprocessor system, containing q processors. For synchronous parallelization, we use a completely connected network of q processors, and distinguish the processor (*master*) that communicates with the user, usually marked as processor 0. The other $q-1$ processors are called *working processors* (*slaves*), indexed from processor 1, up to processor $q-1$. A completely connected topology containing $q=5$ processors is shown in Fig. 6.3(a). The synchronous parallel versions of BCO execute on all q processors, i.e., the computations are assigned to the master as well. In our experiments, we use a different number of processors, ranging from 2 to 20.

In the first variant of GBCO (with centralized information exchange) we require a central blackboard, i.e., global memory. Under the MPI implementation we need to simulate it in such a way that each processor would be able to put its current best solution into global memory, and to read the actual global best from it, without disturbing the others. This situation is realized on the master-slave architecture (Fig. 6.3(b)). Each slave is connected only to the master and communicates with it. The master serves as the global memory, which means that it does not perform any calculations connected to the execution of BCO. The primary role of the master is to manage the global best



(a) Complete interconnection network of five processors.

(b) Master-slave architecture.

(c) Unidirectional processor ring.

Figure 6.3: Network Topology.

solution. It collects improved solutions coming from the slaves one by one, compares them with the current global best, and performs updates if necessary. If no improvement is reported, but a request for a new best solution arrives from a slave, the master sends the current global best solution to that slave. On the other hand, the slaves do not require the current global best as long as they are able to improve their own best solution. They simply report their improvement to the master and continue searching.

In the recent literature [Jov09, Man06, Sev07, Yu11] a non-centralized approach to the parallelization of meta-heuristic methods was proposed. It was implemented via a processor ring architecture. The authors tested uni- and bi-directional cases, and concluded that a unidirectional ring architecture (Fig. 6.3(c)) performs better. Therefore, we also adopted this architecture for the implementation of our non-centralized GBCO. Each processor performs a single iteration of the BCO algorithm and, in the case of an improvement, sends its current global best solution to its predecessor. It also collects potentially improved solution from its successor, compares it with its own current best solution, and adopts the better one as the new reference point. At the end of the execution (when the stopping criterion is met), all current global best solutions are collected and the best among them is reported to the user. In this case, no physical spaces for

the blackboards are needed; they are simulated by the messages used for transferring data between two adjacent processors.

6.4.2 Test instances

For the experimental evaluation, we have chosen a representative subset of test examples from [Dav12], containing two hard instances. The first instance consisting of 100 tasks to be scheduled on 12 machines, Iogra100_12 from [Dav06b] with an *a priori* known optimal solution. This instance can be found at `http://www.mi.sanu.ac.rs/~tanjad/IndepExamples.htm`. The optimal schedule length for the Iogra100_12 instance is 800.

Due to a lack of official benchmark examples for the scheduling problem and the similarities between this problem and the bin packing problem (BPP), the second instance that we use is u250_04 from the bin packing library [Fal96] available at `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html`.

BPP is defined in section 2.3.2 (pg. 29) as the problem of packing a given set $I = \{1, 2, \ldots, n\}$ of n items into minimum number of bins of a fixed capacity C. For each item i its size (l_i) is given. If the items are viewed as tasks and the bins as machines, then the scheduling problem can be seen as equivalent to a BPP with the goal of finding a solution such that the number of bins equals the number of available machines m and the makespan does not increase the bin capacity C. In [Dav12] it was shown that the BCO best solution required two more machines (bins) than the best known BPP solution from literature. In our experiments, we simply report the values of makespan obtained in the best BCO solution for the BPP. Therefore, for the BPP instance u250_04 we assume scheduling 250 tasks to 103 machines.

6.4.3 Comparison of independent BCOc executions

Since it was shown that the first two variants of independent runs of several BCOc algorithms with different seeds provide an excellent speedup, with only a small degradation in the quality of the final solution [Dav11b], we do not reproduce these results here. Instead, we first test our coarse-grained parallelization strategy, named multiple BCOc (MBCO). In order to select the appropriate values for the BCOc parameters (B and NC), we performed numerous sequential executions on the Iogra100_12 example and identified the following best performing combinations:

$$\begin{array}{lll} B = 5, NC = 10; & B = 10, NC = 5; & B = 10, NC = 10; \\ B = 15, NC = 10; & B = 15, NC = 15; & B = 20, NC = 10. \end{array}$$

We order all the variants by their performance and assign them in a cyclic manner to the processors engaged in the parallel execution of MBCO. As the best performing sequential execution, we take $B = 10, NC = 10$, and its results serve as baseline values. Table 6.1 contains the scheduling results for both test instances Iogra100_12 and u250_04. For all examples, the stopping criterion is set to 60 seconds of CPU time.

In the first column of Table 6.1 the number of parallel processors q, executing MBCO, is given. The second column contains the minimum value of the objective function

Table 6.1: MBCO Scheduling results – test problems Iogra100_12 and u250_4.

	Iogra100_12				u250_4			
q	min. MBCO	max. MBCO	av. MBCO	av. CPU time	min. MBCO	max. MBCO	av. MBCO	av. CPU time
1	809	813	811	60.00	149	150	149.7	60.00
2	808	812	811	30.03	149	150	149.8	30.04
3	810	813	811	20.03	149	150	149.7	20.05
4	809	813	811	15.02	150	150	150.0	15.06
5	810	814	812	12.02	150	150	150.0	12.05
6	811	813	812	10.03	149	150	149.8	10.06
7	811	814	813	8.59	149	150	149.9	8.62
8	811	814	813	7.52	150	150	150.0	7.56
9	812	814	813	6.68	150	150	150.0	6.71
10	809	815	813	6.02	150	150	150.0	6.05
11	811	815	814	5.48	150	150	150.0	5.50
12	811	816	814	5.03	150	191	154.1	5.06
13	811	816	814	4.65	150	197	163.6	4.67
14	812	815	814	4.31	150	193	164.0	4.35
15	810	816	814	4.02	150	208	174.2	4.05
16	810	817	815	3.77	150	234	179.6	3.83
17	813	818	815	3.56	150	239	181.9	3.60
18	812	816	814	3.35	169	225	199.4	3.41
19	814	817	816	3.19	181	261	215.0	3.25
20	814	817	816	3.03	181	260	221.9	3.08
av.	811	815	813.42	8.23	154.05	184.63	165.94	8.26

(schedule length), out of 10 repetitions. The corresponding maximum value is presented in the third column of the table. The average schedule length (rounded to the nearest integer value for Iogra100_12) represents the content of column four, while the average maximum over q MBCOs' running (CPU) times are placed in the fifth column. Namely, it is important to note that the CPU time, required by MBCO to complete the necessary computations, is actually the CPU time of the processor which is the last to finish its work. Accordingly, it is equal to the maximum overall running time of the processors. The only difference between two groups of the results is that for u250_4 the average schedule lengths are not rounded to the nearest integer value.

From the results presented in Table 6.1 we can conclude that by engaging different variants of the BCO algorithm the quality of the solution for Iogra100_12 is preserved up to the $q = 5$ within a proportionally smaller wall-clock time. For the second test example (u250_4), the solution quality is preserved for $q < 12$, but later deviations are quite large, going to almost 50% for $q = 20$. As the number of engaged processors increases, the benefit of parameter variations is nullified by the reduction of the CPU time allowed to each BCO. According to the results from Table 6.1, we conclude that, when MBCO is executed, the linear speedup pertains the (best) solution quality for both test instances if $q \leq 5$.

6.4.4 Comparison of cooperative executions

6.4.4.1 Comparison of CBCOs

For synchronous cooperation, we select the same six instances of the BCOc algorithm, and assign them to the processors in the same cyclic manner. We allow our parallel BCO, regardless of the number of engaged processors, to run for the same amount of CPU time as the sequential BCOc, and check for an improvement in the quality of the final solution. During the cooperative execution, different instances of the BCOc algorithm exchange their best solutions at predefined communication points, and the best among these solutions is used to guide further search for all BCOc variants. We implement two versions of CBCO, one with a fixed (10 in our case) number of communications, and the other with processor-dependent number of communications. The obtained results are presented in Tables 6.2 and 6.3.

Table 6.2: CBCO Scheduling results – test problem Iogra100_12.

	Variable number of communications				Fixed number of communications			
q	min. CBCO	max. CBCO	av. CBCO	av. min time	min. CBCO	max. CBCO	av. CBCO	av. min time
1	809	813	811.5	37.90	809	813	811.5	37.90
2	809	812	810.5	45.57	809	812	810.6	30.45
3	809	812	810.6	27.71	809	812	810.1	28.70
4	811	813	811.9	30.78	809	811	810.4	33.51
5	810	813	811.3	31.74	808	811	810.0	39.68
6	809	813	811.2	28.12	809	811	810.4	36.00
7	809	812	810.6	31.43	808	811	809.7	31.38
8	810	813	811.2	33.45	809	811	810.2	26.68
9	807	812	809.9	38.02	809	811	810.5	37.73
10	807	813	810.1	31.79	809	812	810.7	22.96
11	807	812	810.0	30.52	809	812	810.5	26.62
12	809	812	810.9	32.13	810	812	811.1	22.38
13	808	812	810.6	40.28	810	812	811.3	21.40
14	808	813	810.8	34.69	808	813	810.8	21.42
15	810	815	812.2	30.29	810	812	810.8	30.41
16	810	814	812.1	39.69	811	813	811.6	25.91
17	810	813	811.6	45.89	809	812	810.5	31.88
18	810	814	811.8	39.99	809	812	810.6	27.86
19	807	814	810.8	32.78	806	813	811.0	25.37
20	811	815	812.8	30.53	810	813	811.5	34.39
av.	809.00	813.00	811.10	34.49	809.00	811.89	810.65	29.20

Table 6.2 is divided into two parts: the left part is designated as the variant of CBCO in which the number of communications increases with the number of engaged processors, while analogous results, in the case of the fixed number of communications, are given in the right part of the table. We perform 10 repetitions of both variants of CBCO, and in Table 6.2 present the minimum, maximum, and average obtained objective function values, as well as the average (over ten executions) CPU time required to obtain the best solution. As in previous case, the average values are rounded to the nearest integer. As can be seen in Table 6.2, in both cases the quality of the obtained final solution is either improved or at least preserved for $q \leq 12$, with respect to the

sequential execution of the best performing BCOc. At the same time, in the majority of cases, the CPU time is reduced. We cannot expect complete scalability because of the highly stochastic nature of the BCO algorithm. Another interesting conclusion is that the time required for communication and synchronization between processors can be neglected, as the largest value is less than $1s$ (comparing to the total allowed CPU time of $60s$). As Table 6.2 shows, the CBCO variant with less frequent communications gives a slightly better performance with respect to both solution quality and minimum CPU time.

To illustrate the benefits of using CBCO, in Fig. 6.4 we present the improvement of the solution quality obtained for $q \leq 5$ over time. The results of a single run for the variant with a fixed number of communications are shown. The vertical lines in Fig. 6.4 are to indicate a proportionally smaller execution time. As can be seen from Fig. 6.4 parallel executions significantly outperform the best sequential BCOc within a proportionally shorter execution time. Moreover, at the end of parallel executions better final solutions are obtained, indicating that within the same wall-clock time we are able to gain in the solution quality. However, the graphics intersect due to the stochastic nature of BCOc. Comparing different parallel executions, we can conclude that all parallel executions exhibit rapid improvements at the beginning of execution. Later on, some of the executions (for $q = 4, 5$) are continually making small progress during the entire scanned period, while others (for $q = 2, 3$) show a stagnation.

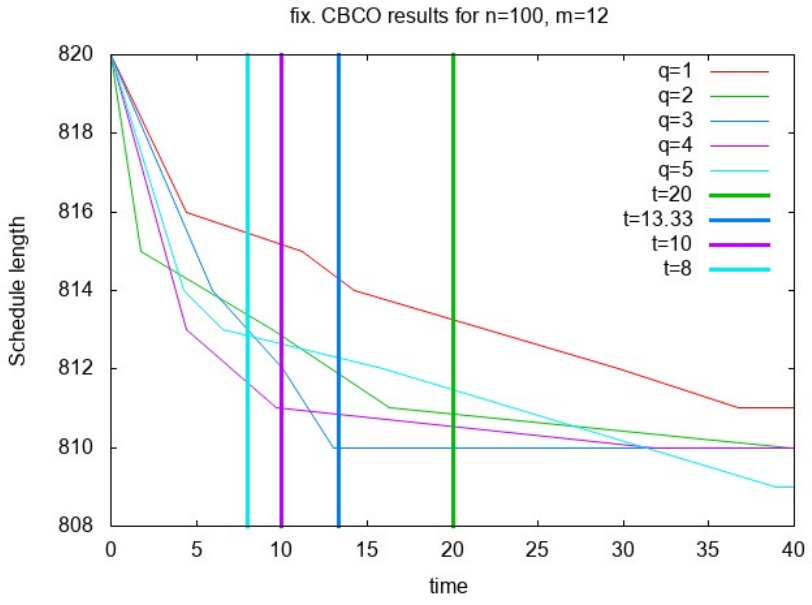


Figure 6.4: Improvement in time of the current best solution, during CBCO with a fixed number of communications

The similar graphic for CBCO with a processor-dependent number of communications is provided in Fig. 6.5. As can be seen, for each q, the solution obtained in $40/q\,s$ is better than the corresponding solution obtained by the best sequential BCO. Moreover, the solution quality stays better, or at least equal to the sequentially obtained one, until the end of the execution. This completely coincides with the results presented

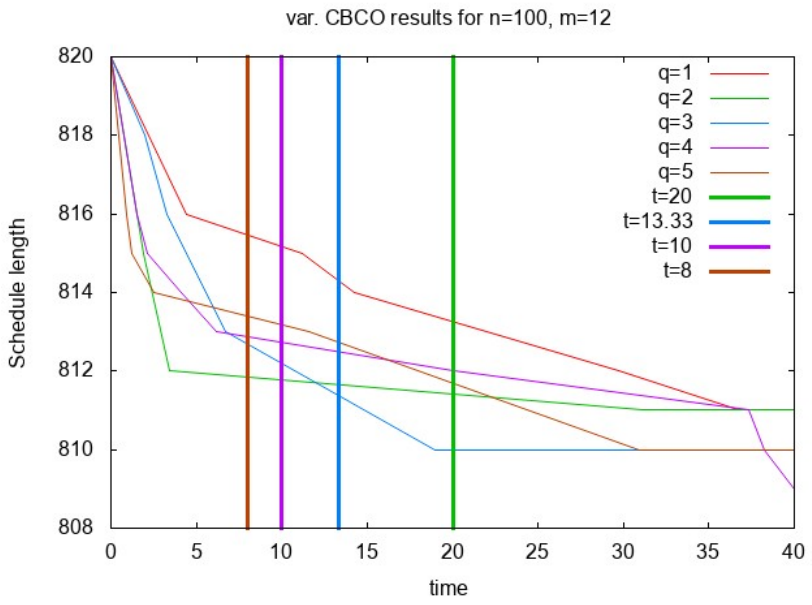


Figure 6.5: Improvement in time of the current best solution, during CBCO with a variable number of communications

in Table 6.2. Comparing different parallel executions, we can conclude that some of the executions (for $q = 3, 4$) are continually making small progress during the entire scanned period. On the other hand, for $q = 2, 5$ we identify a rapid improvement in the quality of the solution at the beginning of the execution, which is later outperformed by the executions on $q = 3, 4$ processors. Other possibilities (degradation of the final solution, improvements over the sequential execution within more than $t/q\,s$, etc.) did not occur in these examples.

The results of CBCO applied to the u250_04 example are presented in Table 6.3. This table has the same structure as Table 6.2. It shows that CBCO with the fixed number of communications performs better: the quality of the solutions or minimum running time are improved.

6.4.4.2 Comparison of GBCOs

The results of asynchronous parallel BCOc executions are presented in Tables 6.4 and 6.5. These tables have the same structure as the previous ones. As can be seen from these tables, for the Iogra100_12 example on average the processor ring shows a slightly better performance regarding the quality of the solution, while the master-slave architecture appears to be a little bit faster.

With respect to the synchronous execution, when comparing the results from Tables 6.2 and 6.4 we can conclude that for smaller values of q, CBCO performs better regarding the solution quality and starting with $q = 14$ GBCO shows better performance. However, considering the average execution time CBCO with fixed number of communications outperforms all other cooperative executions. The presented results for the u250_4 example show that non-centralized asynchronous execution (GBCO ex-

Table 6.3: CBCO Scheduling results – test problem u250_04.

	Variable number of communications				Fixed number of communications			
q	min. CBCO	max. CBCO	av. CBCO	av. min time	min. CBCO	max. CBCO	av. CBCO	av. min time
1	149	150	149.7	16.20	149	150	149.7	16.20
2	149	150	149.9	6.59	149	150	149.8	8.39
3	149	150	149.7	13.59	149	150	149.5	19.32
4	149	150	149.7	13.53	149	150	149.5	19.63
5	149	150	149.8	7.12	149	150	149.5	18.75
6	149	150	149.6	16.95	149	150	149.8	8.8
7	149	150	149.7	12.17	149	150	149.6	14.86
8	149	150	149.8	9.85	149	150	149.4	20.04
9	149	150	149.7	11.38	149	150	149.3	22.43
10	149	150	149.8	12.43	149	150	149.2	23.6
11	149	150	149.9	8.21	149	150	149.3	25.12
12	149	150	149.8	8.79	149	150	149.4	22.62
13	149	150	149.9	8.96	149	150	149.8	11.14
14	149	150	149.9	11.09	149	150	149.9	6.29
15	149	150	149.7	18.97	149	150	149.7	12.06
16	149	150	149.9	10.74	149	150	149.4	25.21
17	149	150	149.9	10.69	149	150	149.7	18.3
18	149	150	149.9	10.06	149	150	149.9	5.15
19	150	150	150.0	12.68	150	150	150.0	5.81
20	149	150	149.9	15.81	149	150	149.8	10.53
av.	149.05	150.00	149.82	11.56	149.05	150.00	149.61	15.69

ecuted on a processor ring) outperforms on average all the other parallel variants with respect to the solution quality, while CBCO with variable number of communications is the fastest variant.

6.4.4.3 Comparison of sequential and cooperative BCOs

To analyze the resource utilization we perform an additional experiment. Instead of using the best performing sequential BCOc algorithm, we implement the version that cyclically changes values of BCOc parameters (B and NC) using the same six combinations as the parallel BCOc versions. We let this sequential algorithm to run for proportionally longer time and compare it with the best performing parallel variants running 60 seconds on the corresponding number of processors. In such a way we equalize the total CPU time of parallel and the corresponding sequential execution. The obtained results are summarized in Table 6.6.

The first column of Table 6.6 contains the total CPU time (the multiple of $60sec$) allowed for the sequential BCOc. Average (over 10 executions) of schedule lengths obtained within the given time limit and the corresponding average minimum times for sequential BCOc are given in columns two and three, respectively. Column four contains the number of processors for parallel executions lasting $60sec$ of wall-clock time. The next two columns contain average schedule lengths and average minimum time for better among cooperative BCOc algorithms, while the remaining two columns are devoted to non-centralized GBCO. As it is expected, the sequential results improve systematically with the increasing in the allowed running time. On the other hand, the

minimum time required to realize these improvements also increases. The summary results in the last row of Table 6.6 show that in order to improve the average schedule lengths for only 0.2% by the sequential BCOc we would have to wait more than 10 times longer. Therefore, our conclusion is that the application of parallel BCOc is more than justified for the considered problem.

6.5 Final remarks

The BCO method is suitable for parallelization as it operates on a population of solutions by utilizing stochastic, either constructive or improvement, heuristic methods. The main concept was introduction of Multiple BCO (MBCO), where we vary the values of the BCO parameters and change the stopping criterion at the same time. We consider three strategies for parallelization of BCOc and propose five coarse-grained parallel implementations under the Message Passing Interface (MPI). The first strategy assumes independent execution of various BCO algorithms. Sequential versions of BCOc are executed on different processors and the best solution is collected at the end. Applying independent parallelization of MBCO to $P||C_{max}$ we obtained almost linear speedup for a modest number of engaged processors (≤ 12). At the same time quality of the solution is not degraded significantly (below 3% with respect to the sequential result).

The second strategy is related to synchronous cooperative execution of various BCOc

Table 6.4: GBCO Scheduling results – test problem Iogra100_12.

	Master-slave architecture				Processor ring architecture			
q	min. GBCO	max. GBCO	av. GBCO	av. min time	min. GBCO	max. GBCO	av. GBCO	av. min time
1	809	813	811.5	37.90	809	813	811.5	37.90
2	809	812	811.1	37.46	808	813	811.0	19.14
3	809	812	810.5	35.88	809	812	810.8	24.45
4	808	812	810.8	30.00	810	813	811.1	35.52
5	809	812	810.9	34.37	808	812	811.0	36.01
6	809	812	810.5	37.14	809	812	810.8	29.09
7	810	812	810.8	26.32	810	812	811.1	39.14
8	810	813	811.4	25.88	809	812	810.7	36.01
9	809	812	810.4	34.80	809	812	810.8	36.95
10	808	813	810.9	34.10	808	811	809.8	42.61
11	807	813	811.1	25.32	809	812	810.4	38.10
12	809	813	811.3	35.15	809	812	810.5	36.88
13	810	813	811.3	33.18	809	812	810.9	36.10
14	807	812	810.9	25.08	809	812	810.4	42.53
15	809	812	810.7	30.81	809	812	810.7	30.14
16	810	812	811.0	31.40	808	812	810.5	32.47
17	807	812	810.8	22.60	810	812	810.8	32.76
18	809	812	810.8	33.42	810	812	810.6	35.75
19	810	813	811.1	28.32	809	813	811.0	35.29
20	809	813	811.3	16.58	809	812	810.7	31.61
av.	808.84	812.37	810.93	30.41	809.00	812.11	810.72	34.24

Table 6.5: GBCO Scheduling results – test problem u250_04.

	Master-slave architecture				Processor ring architecture			
q	min. GBCO	max. GBCO	av. GBCO	av. min time	min. GBCO	max. GBCO	av. GBCO	av. min time
1	149	150	149.7	16.20	149	150	149.7	16.20
2	149	150	149.8	9.93	149	150	149.8	14.42
3	149	150	149.7	12.33	149	150	149.5	16.25
4	149	150	149.6	14.01	149	150	149.5	20.65
5	149	150	149.8	6.40	149	150	149.7	15.49
6	149	150	149.5	9.50	149	150	149.6	17.32
7	149	150	149.4	21.17	149	150	149.6	12.08
8	149	150	149.3	16.86	149	150	149.4	15.49
9	149	150	149.4	23.75	149	150	149.7	14.49
10	149	150	149.3	25.54	149	150	149.5	17.11
11	149	150	149.2	27.00	149	150	149.5	20.9
12	149	150	149.2	19.93	149	150	149.2	29.87
13	149	150	149.4	17.17	149	150	149.3	27.42
14	149	150	149.6	12.42	149	150	149.3	27.98
15	149	150	149.4	20.89	149	150	149.2	31.25
16	149	150	149.4	16.92	149	150	149.3	34.96
17	149	150	149.5	21.56	149	150	149.1	32.26
18	149	150	149.6	11.65	149	150	149.2	31.36
19	149	150	149.5	17.09	149	150	149.5	20.98
20	149	150	149.3	32.17	149	150	149.3	32.42
av.	149.00	150.00	149.47	17.70	149	150	149.43	22.77

instances. We implemented two synchronous cooperative variants on a completely connected homogeneous multiprocessor system, in which processors communicate by exchanging messages. The one involving a less frequent knowledge exchange resulted in better performance for the considered MBCO.

Finally, related to the third strategy, we implemented two variants of asynchronous BCO parallelization. The first of them includes a global memory concept, and it is implemented on master-slave multiprocessor architecture. The second, a non-centralized asynchronous execution, is realized on a unidirectional processor ring. To the best of our knowledge, these are the first implementations of asynchronous parallel bee-inspired meta-heuristic methods.

On two hard test examples we showed that, while both the synchronous and asynchronous concepts perform well on a modest number of processors, the asynchronous concept outperforms the synchronous one as the number of engaged processors increases. Therefore, our main contribution is the successful development of new and efficient distributed memory parallelization strategies for BCO.

6.6 Chapter summary

The chapter begins with an overview of parallelization strategies, where we discuss about the goals of parallelization of meta-heuristic. Then, we provide detailed descriptions of parallel implementation of two nature-inspired meta-heuristics (ABC and

Table 6.6: Comparison between sequential and parallel BCOc within the same CPU time – test problem Iogra100_12.

Total CPU time	Sequential BCO			CBCO-Fixed comm.		GBCO-Processor ring	
		av.			av.		av.
$(q * 60sec)$	av. SL	min time	q	av. SL	min time	av. SL	min time
120	810.7	60.27	2	810.6	30.45	811.0	19.14
180	810.3	86.69	3	810.1	28.70	810.8	24.45
240	809.8	126.50	4	810.4	33.51	811.1	35.52
300	809.8	126.50	5	810.0	39.68	811.0	36.01
360	809.6	176.61	6	810.4	36.00	810.8	29.09
420	809.6	176.61	7	809.7	31.38	811.1	39.14
480	809.4	231.51	8	810.2	26.68	810.7	36.01
540	809.2	283.49	9	810.5	37.73	810.8	36.95
600	809.0	320.66	10	810.7	22.96	809.8	42.61
660	808.8	332.44	11	810.5	26.62	810.4	38.1
720	808.7	357.47	12	811.1	22.38	810.5	36.88
780	808.5	417.83	13	811.3	21.40	810.9	36.10
840	808.5	417.83	14	810.8	21.42	810.4	42.53
900	808.3	496.91	15	810.8	30.41	810.7	30.14
960	808.3	496.91	16	811.6	25.91	810.5	32.47
1020	808.3	496.91	17	810.5	31.88	810.8	32.76
1080	808.2	517.00	18	810.6	27.86	810.6	35.75
1140	808.2	517.00	19	811.0	25.37	811.0	35.29
1200	808.2	517.00	20	811.5	34.39	810.7	31.61
av.	809.02	324.01		810.65	29.20	810.72	34.24

ACO). The survey shows that their parallel implementations exhibit good performance, which was the inspiration to conduct similar approaches on BCO.

In the second part of this chapter we propose parallelization strategies for BCO and have tested them on BCOc implementation for $P||C_{max}$. We conclude that cooperative approach is more powerful than independent executions, as it was expected. Among cooperative strategies, we compared synchronous and asynchronous and confirmed that increasing number of processors over some threshold makes it difficult to synchronize processors. In these cases asynchronous parallelization strategy manifests better performance.

CHAPTER 7

Methodology of experimental study of BCO

In the last decade the BCO community has encouraged systematic experimental research either as an integral part of the BCO development and/or during experimental evaluation of particular implementations. In this chapter we aim to contribute to further development of methodological study of BCO and explore different combinations of the BCO parameters' configurations by addressing the question of parameter control (influence of parameter values). Numerous strategies to experimental study of meta-heuristics have been proposed in the literature. For the most part, the strategies are efficient as they try to minimize the overall computational coast. However, the choice of an appropriate automatic tuner can often be time consuming. In addition, integrated statistical and visual tests might not address the goals of a particular study. We opt for statistical and visual tests that help establish conclusions after collection of measured outcomes.

The chapter is organized as follows. We first review tutorials regrading experimental analysis of meta-heuristics. Various aspects of the experimental methodology are described, along with discussions and literature surveys. The objective is to show how a research goal might influence the selection of a performance measure and configuration method. To distinguish it from the definition in the field of statistics, we review definition of *sensitivity analysis* applied in optimization. In this chapter we emphasize the importance of reproducibility as a contributor to every unbiased study. Due to their broad utilization, we identify and describe the most frequently employed statistical tests in OR. In the last section we focus on the subject of the BCO parameters and their interdependence. We introduce new concepts regarding different components of the BCO algorithm and reference them as new BCO's parameters. The parameters are categorized and their interdependence described by a *hierarchy diagram*.

7.1 Motivation for empirical analysis

Empirical analysis is a study based on data collected by observations or experiments. *Experimental analysis* of meta-heuristic methods is occasionally employed as a synonym [Kao08, pg. 290], therefore, we make no distinction and we use them interchangeably throughout the dissertation. Empirical analysis of meta-heuristics in many cases requires great effort due to the stochastic nature of the method and/or parameter interactions.

The aim of this dissertation is to provide insights into the performance of BCO by following the guidelines of Hooker [Hoo94, Hoo95], Barr [Bar95], Rardin [Rar01] and Johnson [Joh02b], who together with many other researchers [McG96, BB06], were concerned with methodical analysis of heuristic and meta-heuristic methods. Meta-heuristics are characterized by their parameters and/or different algorithmic components [Bir09]. Once implemented, the algorithm can exhibit different performance as parameter values change. The awareness of how to conduct a comprehensive empirical study of meta-heuristics has increased over the last decade in the operations research community [Stü09]. There are various recommendations in the literature, however, identifying the best one is an ongoing topic of interdisciplinary research.

In [Hoo95] Hooker emphasizes the importance of thorough experimental analysis, as opposed to *competitive testing* focused solely on the design of heuristic algorithms for a specific instance or a class of problems for the purpose of outperforming a state-of-the-art method. Utilization of the so-called *controlled experimentation* might be the only way to find out why some heuristic procedure has demonstrated a good performance. Therefore, embracing the terminology of Hooker, the subject of the thesis is *scientific testing* of the BCO method. By scientific testing we assume that we plan to discover the method's properties by learning, for example, how the algorithmic components and/or properties of problem instances effect algorithm's behavior. The most widespread variety of empirical study in OR and computer science concerns configuration (tuning) of the algorithm. It is clear that obtaining the best configuration is an optimization problem itself because of a large number of algorithm's parameters, or merely due to the size of the parameters' domain. Several other proposals of well-planned and extensive testing of heuristics have been given by Barr *et.al* [Bar95], McGeoch [McG96], Bartz-Beielstein [BB06], Birattari [Bir09], just to mention a few.

7.2 Experimental study of meta-heuristics

7.2.1 What is an experiment?

An *experiment* represents a collection of independent runs of an investigated algorithm. A *run* of the algorithm corresponds to a single execution, occurring through iterations, until a stopping criterion is satisfied. In particular, an experiment consists of n_{run} different runs of the algorithm's instance (each specified by the fixed value of the random number generator, $seed$) conducted under the same conditions. The conditions are determined by the algorithm's components, such as heuristic rules for construction/-modification of a solution, population size or an operator. The number of independent runs of one experiment is often, in the literature, appointed to 100 in order to assure statistical significance when comparing different data samples [Hoo07, Blu08].

7.2.2 Measure of performance

A stochastic nature of meta-heuristics requires executions of the algorithm several times in order to capture the central tendencies of the response values. A response value commonly refers to the quality of a solution (sometimes referred to as usefulness or utility), or the computational effort (e.g., running time). There are different ways to define

measure of the algorithm's performance, depending on the goal of the research. For instance, if the idea is to capture performance of an algorithm under limited amount of time (or number of evaluations) then classical tools, such as mean and standard deviation, are good representatives of the performance measures. This kind of estimates provide what is known as *effectiveness* of the algorithm [Bar95, BB06, Bar11]. However, measures of effectiveness can be restricted, especially when a distribution of the response variable does not follow any bell or symmetric shape (e.g., Gaussian). If the idea is to estimate an effort to reach a solution of a predetermined quality, the performance measure describes what is known as *efficiency* of the algorithm. In addition, a mixed measure of the two is possible [Bar11]. The intention of our study is to inspect the effectiveness of proposed BCO algorithms. We believe that the gathered knowledge might be used to determine the efficiency of the BCO algorithm for different test instances. A thorough review of different performance measures is provided in [BB06, pg. 110] and in [Bar11, pg. 40].

Here, we shortly describe several types of performance measures suggested in the literature in order to discuss various aspects of the study. For example, organized competitions between stochastic algorithms imply restrictions on the total running times. Conversely, strict limitations are usually not imposed while researchers try to understand the mechanisms that led towards a certain outcome.

We begin with 2010's competition "Real-Parameter Black-Box Optimization Benchmarking" (BBOB-2010), between various heuristic algorithms for continuous optimization problems [Han10a]. The requirement of the competition was a small value for n_{run} (set to 15). The explanation is: if the number is set even a little bit higher, then the irrelevant performance differences become statistically significant [Han10a]. The second requirement was the target value $f_{target} = f_{opt} + 10^{-8}$. The advocated measure of performance was *expected running time*, ERT, introduced in [Hoo98b]. ERT represents the expected number of function evaluations until reaching f_{target} for the first time. It is determined as a ratio of the number of *unsuccessful evaluations* [(1)] (in all runs of one experiment) and the number of *successful* runs [Han10a, BB13]. More precisely,

$$ERT(f_{target}) = \frac{\#FEs(f_{best}(FE) \geq f_{target})}{\#succ}.$$

The authors of [Han10a] advice to employ ERT because it is ideal for wide variations and ratio scales, simple and easy to interpret.

Another performance measure, commonly used for stochastic algorithms with high variability, is presented by Hoose and Stützle [Hoo98b]. It is known as *run-time distribution* (RTD), or more generally, *run-length distribution* (RLD). The authors have demonstrated its usefulness on the Las Vegas algorithm. To attain RLD the algorithm is executed n_{run} times for a longer period of time, i.e., until reaching an optimal or some other target solution. During the experiment the time to obtain the target solution is measured and reported. If $rl(j)$ is the number of steps that leads to successful run of the jth execution of the algorithm, then the cumulative empirical distribution of the RLD is defined as

$$P(rl \leq i) = \frac{\#j|rl(j) \leq i}{n_{run}}.$$

[(1)] Unsuccessful evaluation is considered to be the one where the observed objective function is worse then a given target [BB13].

According to the literature, selection of a performance measure may depend on the problem to be solved. In fact, the criteria by which one is favored over the other is not yet clear [Bar11, pg. 38]. Furthermore, if the problem is too hard for a stochastic algorithm to find an optimum, or if there is no previous knowledge about the quality of a solution, then utilization of descriptive statistics (such as mean or median) is appropriate [Eib02].

The performance measures featured by central tendencies are employed in this thesis, i.e., average value of a solution quality (or relative error) and average value of running times (number of evaluations). We have opted for this approach to establish optimal parameter configurations (as they are problem dependent). In addition, our study is focused on observations and conclusions about the variability of outcomes that the BCO algorithm exhibits over different settings of its parameters/modules.

7.2.3 Configuration methods

Tuning of an algorithm is a process of configuring meta-heuristics in order to discover configuration of the method's parameters which enables the best possible performance for a particular problem instance [Bir09]. Numerous doctoral thesis, especially published in the recent years, are devoted to the topic of configuration (tuning and sampling) of meta-heuristic methods [DJ75, Bes04, Rid07, She11, Bar11, Smi12, EO13, Lai14]. Various research papers pursue the same questions [Gre86, Bir06, Eib11, Hoo11, Les11, Hut11, BB13, LI14], proposing and/or inspecting different tools for experimental analysis. The tools are designed to avoid analysis of algorithms' performance on the complete parameter space. Among the first is *F-Race* as a representative of *racing algorithms* [Mar97, Bir09]. The procedure is based on *re-sampling* techniques such as *bootstrap* in order to avoid multiple runs while pertaining statistical significance of conclusions [Bir09, pg. 5]. *F-Race* relies on the non-parametric Friedman's two-way analysis to compare different parameter configurations. Another racing method is iterated PARAMILS [Hut07, LI14]. Both approaches are known as *model-free* algorithm configuration methods because no assumption about the response surface landscape is made. On the contrary, *model-based* methods are iterative procedures founded on model fitting of the response surface in order to attain a good set of parameters' values for the next iteration [Hut10b]. Design of Experiments (DOE) [Win62, Mon01] represents an integral part of the model-based methods. A thorough introduction to DOE, adjusted for performance analysis of stochastic algorithms, is given in [BB06] and in [Rid07] it was successfully applied to tuning of ACO parameters for TSP. Another novel method for analyzing and tuning parametrized stochastic algorithms is Sequential Parameter Optimization (SPO) toolbox [BB13]. SPO is based on successive improvements by estimating relationships between parameters of a method and the corresponding response values. The machine-learning aspect of the tuning process is covered in [Bat08], whereas [Les11] presents procedures inspired by data mining. [Hut10b, Hut11] propose SMAC to overcome the limitations of sequential model-based optimization methods focused on configuration of deterministic algorithms. SMAC is based on *empirical hardness models* (EHMs) that estimate the runtime of an algorithm, and thus it selects the promising configurations explored in the next iteration [LB14]. The majority of today's tuners are model-based and employ linear or nonlinear models or other sophisticated approaches, such as quadratic ridge regression, neural networks,

regression trees, random forests, etc [LB14]. In general, majority of parameter tuners follow three basic principles: screening, experimentation and exploitation. The literature on the subject of tunning is overwhelming, therefore, the correct choice of the configuration method for BCO remains one of the future challenges. A comprehensive classification of the state-of-the-art automatic tuners has been provided in [Eib11].

We distinguish two broad classes of tuning: *off-line* and *on-line*. Off-line implies that assigning values to all the algorithm's parameters is conducted at the beginning of an experiment. The idea behind on-line tuning is obtaining the feedback, which is used to modify the parameters while the search itself is still in progress [Bir09]. For both classes it is suggested that two independent sets of problem instances should be treated: *tuning instances* and *test instances* [Bir09]. Tuning instances are applied to find the best-performing configuration of the algorithm parameters. Once the parameters are found, the algorithm can be evaluated on a different set of instances, indicated as test instances. In this way an unbiased estimate of the algorithm's performance can be conducted.

7.2.4 Types of parameters

From the field of statistics we define a *factor* or *parameter* as any controllable variable which could affect the outcome or the result of an experiment. Two types of parameters are distinguished [Win62, Eib11]: *qualitative* and *quantitative*. Quantitative parameters are mostly numerical values with sensible ordering. Qualitative parameters represent symbolical values that are difficult to measure, e.g., evaluation functions. The specifications are similar to those introduced in [Bir10, pg. 338] where two types of parameters are distinguished: *numerical* and *categorical*[(2)]. Categorical parameters are defined as: "*different procedures or discrete choices that can be taken by an algorithm*" [Bir10]. Therefore, qualitative factors are variables used to represent categorical data, taking values that are names or labels to substitute expressions or formulas. A different terminology exists concerning the types of parameters, one that elegantly reflects the difference between two types of algorithm tuning [Bir09, Eib11]. *Structural tuning* is conceived as an analysis of influence of qualitative factors on the performance measure, while the *parameter tuning* is associated with influence of quantitative factors.

7.2.5 Sensitivity analysis

A complete scientific study of the BCO method is overwhelming, and therefore our focus is on the investigation of parameters' influence with the tools of *sensitivity analysis*. Sensitivity analysis in OR and computer science may be considered as a part of the broader field of scientific testing. It is a term used in relation to studies of variance in algorithm's response values while changing the parameter configurations or problem instances [Hoo07]. Research in sensitivity dates back to the development of linear programming, according to [Gal97, Chapter 1]. For general definition of the sensitivity analysis we refer to Definition 11, found in [Nak13, pg. 5]. The author implies that with the lower variability of the obtained solutions we obtain the better sensitivity. A *consistency* of parameter configurations is also an important aspect of sensitivity analysis, i.e., establishing to which degree an impact of parameter configurations is similar

[(2)] Categorical data is one which can have two or more categories, but there is no intrinsic ordering [Cal]

across the benchmark set of problem instances [Hoo07]. We are particularly interested in *parameter sensitivity analysis*[3], as provided in Definition 12. In the study of BCO, we do not conduct exhaustive analysis of interactions between the parameters, and concentrate on the main effects.

Definition 11 (**sensitivity analysis**, [Nak13])**.** Sensitivity analysis of a optimization algorithm corresponds to insensitivity against small deviations in either problem instances or the parameters of the algorithm. ◇

Definition 12 (**parameter sensitivity analysis**, [Hoo07])**.** Sensitivity analysis of a optimization algorithm corresponds to analyzing variation of an algorithm's outcome in response to changes in its parameter settings. ◇

In order to explore hidden properties of an analyzed data the tools of *exploratory data analysis*, such as boxplots and histograms, are typically utilized together with other techniques of visual analysis [Tuk77, Rar01, Joh02b, Cza04, BB04, BB06, Rid07, Bir09, Eib11].

7.2.6 Reproducibility

An important feature of any research presentation needs to consider the factor of *reproducibility* [Bar95, Hoo95]. Namely, the utilization of the random number generator causes systematic variability in the response values. The reason can be found in the methods employed to produce stochastic behavior of the algorithm, thus, influencing all the executions across the study. By *reproducibility* we imply that when repeating the study with the same settings, we generate the same outcome. Therefore, we distinguish parameter $seed$ from others as *nuisance* factor since it is not the subject of the experimentation [Mon01]. It should be controlled during the experimentation and fixed to values that assure reproducibility under the same experimentation conditions [Cza04]. Value of $seed$ might be reported with other descriptive data. Another contribution to the reproducibility of the empirical analysis is to provide as much information on results as possible. Moreover, employing identical random seeds may contribute towards adequate choice of statistical tests focused on exploring an influence of main effects [Hut10a].

7.2.7 Statistical methods

Many research questions can be formulated in form of a statistical hypothesis. Usually, a hypothesis is defined as a statement where we compare means or medians of two data sets or one set with a specified value. A *null hypothesis* is usually defined to propose no difference between compared values. An *alternative hypothesis* would correspond to a statement that proposed values (or data sets) are statistically different. At the beginning of a hypothesis testing it is required to determine a *critical* or *rejection region*. Critical region represents a set of values that helps establish the rejection of the null hypothesis. It is worth noting that statistical tests are not absolute and can produce certain types of error. For example, if a result of a hypothesis test is that we should

[3] The expression *parametric analysis* is commonly employed for the parameters of a optimization problem, also know as parameter optimization [FB07].

reject a null hypothesis while in fact it is true, then *type I* error has occurred. This is why at the beginning of an experiment we need to assure to which degree we are ready to cause the type I error, i.e., with which probability we expect the error to happen. Such probability is known as *significance level* of a test, often denoted by α. Significance level of statistical tests represents an extreme value for so-called p-value that denotes a result of the statistical test. Normally, if p-value is less than α, we reject the null hypothesis and accept the alternative hypothesis. Otherwise, we fail to reject the null hypothesis. This describes a critical value approach to conduct a hypothesis test [Mon01, Leh06, Ric06, Kir07, How10].

The correct choice of a statistical test relies on more factors, such as: sample size (n_e), distribution of data sample under review, dependency between data, etc. A sample can be: original data and/or difference between two data samples. There are two types of hypothesis testing procedures [Cro12]: (a) parametric; and (b) non-parametric. Parametric tests are employed on samples with *interval* or *ratio* [(4)] data which satisfy the condition of normal (Gaussian) distribution or when departures from normality are not substantial. These condition need to be satisfied for different groups of data, like: distribution of an original data or variances of data from the group mean. Each statistical test comes with its own set of conditions. In the case that normality condition is not satisfied, or we are not concerned with the underlying distributions, non-parametric tests may be used [Hol99]. However, the literature recommends the use of parametric tests if the normality condition is satisfied, seeing that non-parametric tests have weak (or non) assumptions about sample distributions.

For data sets with normal distributions, measure of central tendency can be based on the mean and standard deviation of the sample. Otherwise, if the data sample is not normally distributed, then using median and range is advised [BB13]. In the literature, the most common statistical test used to investigate a difference between two algorithms is the t-test [Box05, pg. 39]. There are two types of t-test. If data samples are mutually dependent, *paired t-test* is commonly used. Otherwise, *two independent sample t-test* is employed. Both tests evaluate statistical differences between means of two samples. Comparing multiple samples requires investigating *homogeneity of variance*. If homogeneity of variance is satisfied, most common utilized statistical test is ANOVA test (Analysis of Variance, [Mon01, She04]). It is recommended to conduct post-hoc analysis in order to determine pairs of samples that exhibit statistical difference. ANOVA test is used to establish statistical differences between means of two or more samples. Therefore, ANOVA assumes that a sample follows normal distribution and satisfies a condition of *homoscedasticity*[(5)]. Moreover, parametric tests, such as ANOVA, can be robust to moderate departures from normality, and some general advices are proposed in [Wes95, Kim13]. To make assessment for normality of medium-sized samples ($50 < n_e < 300$), the authors of [Kim13] suggest using z-test (based on skewness and kurtosis) instead formal normality tests, such as Shapiro-Wilk. We follow the first approach during examination of ANOVA conditions.

As ANOVA requires a certain level of homoscedasticity on an independent set of samples, parametric Leven's test is often used [Tal09]. In case we are testing the difference between a variance for paired data, it is recommended to use Spearman correlation

(4) [Ste46] provides a classification of scales of measurements where he introduced four levels: nominal, ordinal, interval and ratio.

(5) Equality of variance.

or less robust Pitman-Morgan test [McC87]. If the p-value of Leven's test is greater than 0.05, we accept a null hypothesis that always assumes an equality of variance. If p-values is smaller than 0.05, there is no proof of homogeneity. It is often the case that the homogeneity test of variance is not needed prior to the choice of analysis method since even conservative test, such as paired t-test and ANOVA, can hold for differences in variances up to four times, or if the largest standard deviation is less than double the smallest standard deviation [Cof00, How10].

Repeated-measures ANOVA, or *single-factor within-subjects analysis of variance*, is used in case of two or more dependent samples with interval/ratio data for which we want to obtain information concerning individual patters of change [Dav02, She04]. For example, when studying effects of a medical treatment over time on the same patients. Furthermore, instead of time change, we can observe the effects of different conditions. Such scenario corresponds to the empirical study, conducted in this thesis, of the BCOc algorithm for the particular scheduling problem. Repeated-measures ANOVA (RMANOVA) presumes normal distribution of data, as well as condition of *sphericity*, analog to the homogeneity of variance for one-way ANOVA test. In `R` package language[(6)] sphericity condition is tested as a part of `ez` package [Law15].

Stochastic nature of meta-heuristic algorithms can often cause a distribution of results to obey some non-Gaussian function. This is why non-parametric tests are common during experimental analysis of meta-heuristics. If two samples of equal size need to be compared, we can choose between [Cof00]: (a) *Wilcox signed rank-sum test* as the analog to the paired t-test; (b) *Wilcoxon-Mann-Whitney U test* as analog to the 2-sample independent t-test; or (c) Kolmogorov-Smirnov. When comparing more than two algorithms we can use: (1) Friedman's non-parametric test which represents analog to an ANOVA with repeated measures; (2) Kruskal-Wallis test, analog to the one way ANOVA. A clearer overview of commonly used statistical tests in the empirical analysis of heuristics is provided in Table 7.1. When data are not normally distributed, or not of equal size, then the non-parametric Levene's test can be used to inspect the homoscedasticity.

Table 7.1: Statistical tests.

#	**Parametric**	**Non-parametric**
Dependent samples		
2	Paired t-test	Wilcoxon signed rank test
> 2	One way ANOVA with repeated measures	Friedman's test
Independent samples		
2	Independent sample t-test	Wilcoxon-Mann-Whitney U test
> 2	One way ANOVA	Kruskal-Wallis

Friedman's test is used for multiple comparison between different samples when we want to investigate a null hypothesis of equivalence of medians. It is based on ranking between samples, and is, therefore, commonly employed to test for differences in samples with ordinal data. However, it can also be utilized for continuous data that has violated assumptions of RANOVA.

[(6)] https://www.r-project.org/

Kruskal-Wallis (KW) test is based on the Wilcoxon-Mann-Whitney (WMW) test for more than two samples. Although Kruskal-Wallis does not assume normality of samples distribution, it does work best when it assumes that the populations have same distributions [Fag09]. A common misunderstanding about a null hypothesis of the Kruskal-Wallis test is that it is often used to examine an equivalence of groups mean or median. Instead, it should be used to test the difference between mean ranks [Fag09]. Kruskal-Wallis can also be described as ANOVA applied to ranks, where ranks are calculated on the overall dataset [Mon01]. However, using a rank transformation can have severe consequences on a test result, especially when two samples do not have identical distributions and equal size [Fag09]. In that case, the rank transformation can change mean, standard deviation and skewness of two samples and therefore should be used when distributions of the two samples are identical and of equal size [Fag09].

Following the test for equivalence of means (or medians) between multiple samples, it is recommended to conduct a *post-hoc* test [Der11]. The choice of a post-hoc test depends on statistical test that was used beforehand. For example, after ANOVA we can employ t-test for pairwise comparison. After a non-parametric tests, we can utilize one of the non-parametric two-sample tests showed in Table 7.1. For example, after Friedman's test, a post-hoc test may employ Wilcox test for multiple pairwise comparison. It is important to note that a multiple pairwise comparison demands a correction for p-value [Cal].

To determine a level of influence of algorithm's parameters, we used additional measure often recommended when conducting parametric test, known as *effect size*. Namely, the measure can quantify a size of an effect of one parameter on the reported outcomes. There are different procedures on how to calculate the effect size, according to [She04]. It is important to point out that a secure way to describe reported values of the effect sizes is usually conducted by comparing them across the study.

7.3 Experimental analysis of BCO

7.3.1 Motivation of the BCO study

Designing a BCO method in principle implies a selection of constructive/improvement moves, an evaluation function and setting BCO parameters to suitable values that are usually determined by pilot studies, previously published work or even intuition. Rigorous considerations of different loyalty probabilities is lacking in the existing literature, despite being an integral component of the generic section of the method and, generally, is not problem specific. In Chapter 4 we have addressed the issue by introducing loyalty functions and argued about their influence on the diversification and exploitation of (partial) solutions. The experimental study conducted in the thesis demonstrate how this properties are influencing the quality of response values. Furthermore, to achieve a good alternative to reported solutions, new evaluation functions are presented and analyzed for two combinatorial problems described in Chapter 2.

In this dissertation we try to address several different goals. One is to accommodate restrictions coming from the real world, such as dealing with a short amount of computer resources in contrast to remaining longer in the working process due to the importance of the quality of a solution. Another objective is to focus on the empirical

analysis of BCO by allowing larger number of repetitions to establish statistical significance. We address this issues by careful consideration of stopping criteria. As the stopping criterion may manifest a large impact on the results of an empirical analysis, two different types are considered for constructive BCO: maximal number of iterations and maximal CPU time. Empirical analysis of the BCOi is conducted by setting a maximal number of objective function evaluations (MAXFLIPS). Therefore, we extend the set of BCO parameters by incorporating stopping criteria as a factor. Stopping criterion represents a central parameter in the literature since, if set too low, it can restrain the heuristic from finding high quality solutions, and if set to high it can waste computational resources. Special care needs to be taken to avoid the possible *ceiling effect* where two algorithms can achieve the maximum level of performance, therefore, should not be compared [Coh95, BB13].

7.3.2 Structure of the BCO study

As previously mentioned, empirical analysis of BCOc and BCOi is conducted by means of an *off-line* systematic adjustments of the parameter values. The approach allows to test robustness of the BCO algorithm to changes in both parameter values and test instances by allocating values before the algorithm execution [DJ75]. Tools of statistical analysis, such as measures of central tendency and (non)parametric statistical tests, are implemented in order to verify the significance of difference between the measured outcomes. To complement or, occasionally, substitute statistical results sensitivity analysis is conducted by visual representation of descriptive statistical data. Total number of independent runs within one experiment (n_{run}) is based on prior work on the problem of scheduling ([Dav12]) and recommendations from the literature [Hoo05, Tal09, Nak13].

In order to conduct practical studies of BCO we consider two well known optimization problems: *static scheduling of independent tasks on identical machines*, annotated as $P||C_{max}$, and the special variant of Boolean Satisfiability problem (SAT), called 3-SAT. In Section 2.3 we briefly described and reviewed some of optimization methods commonly used to deal with these two problems. In the following chapter we investigate the structure of the test instances. For scheduling problem instances we have presented box-plots to demonstrate relation between the problem size and the average value of the tasks' computational times. In the chapter devoted to 3-SAT we review the structure of the corresponding test instances invoking clause-to-variable index. However, the knowledge about test instances is not build into algorithms since their influence is not a subject of our research. The focus is on the algorithmic parts that have the potential to influence positively or negatively the performance of BCO.

Prior to the main experimental studies we had observed that high quality solutions could be obtained if we "reverse" the objective of the problem. Namely, we recognized that instigating maximization of the objective function for the $P||C_{max}$ problem could also lead towards reporting of high quality solutions. This initiated questions about how to depict the corresponding evaluation process. In case of 3-SAT we departure from common analytical definition and introduce an evaluation function as a procedure. This has inspired to establish *method of evaluation* to distinguish between new evaluation strategies. Method of evaluation within the BCOc algorithm is employed to indicate mixture of different objectives. In particular, the objective of the $P||C_{max}$

problem is to minimize the schedule length (*makespan*, i.e., the minimal maximal load of the processor), which we refer to as *minimization principle*. However, under certain restrictions, we can employ *maximization principle*, i.e. , where quality of partial solutions is measured by the maximal value of the *makespan*. Appointing higher quality to solutions that originally have the worst quality, may provide positive flexibility during the construction of interesting results. The approach, however unconventional, might lead the search towards high quality or acceptable solutions. When necessary, the new parameter replaces the evaluation function. The set of all possible methods of evaluation is denoted as $\mathcal{M}$, and its cardinality is problem specific.

7.3.3 Categorizations of BCO parameters

Parameter search space (*configuration space*) of BCO can be, for the most part, described as a product of sets $\{\mathcal{L}, \mathcal{E}/\mathcal{M}, \mathcal{B}, \mathcal{NC}\}$. $\mathcal{L}$ denotes a collection of loyalty functions. $\mathcal{E}$ and $\mathcal{M}$ are collections of all possible evaluation functions and methods of evaluations which might be used interchangeably (their cardinality is problem specific). $\mathcal{B} \subseteq \mathbb{N}$ is a set of countably many elements each indicating a size of a bee population, and $\mathcal{NC} \subset \mathbb{N}$ is a countable set that contains all possible values of parameter NC. The cardinality of $\mathcal{NC}$ may be bounded, e.g., by the number of constructive moves during generation of a complete solutions, while the size of $\mathcal{B}$ is usually determined arbitrary.

BCO parameters can be categorized under the classification from [Eib11] and section 7.2.4 (pg. 129) as qualitative and quantitative. Because of their numerical nature, B and NC represent quantitative parameters. The set of evaluation functions ($\mathcal{E}$) and of loyalty functions ($\mathcal{L}$) may be considered as sets of categorical values. Consecutively, we can establish a distinction between BCOs and the BCO instances. Two BCOs differ if they vary in one of the qualitative parameters. For example, if they implement different loyalty or evaluation functions. When all parameters are specified, an instance of BCO is obtained. The quality of the set of parameters of a BCO instance is denoted as *utility*. For a given instance of a problem, the quality of the reported solution is determined by its evaluation function value and is referred to as a *solution quality*.

7.3.4 Hierarchy diagram of BCO parameters

For the purpose of conducting an empirical study through some meaningful steps, a certain hierarchy of BCO components is established in terms of the design of the algorithm. Hierarchy diagram of the BCO components (modules of the algorithm and/or its parameters) is provided in Fig. 7.1. Diagram reflects importance of particular BCO parameters and therefore helps establish a course of empirical study, particularly statistical analysis. The diagram does not encompass all possibilities of the algorithm design, however, it reflects the most frequent situations. The top node of the diagram represents the most significant component of the BCO algorithm regarding the design because it reflects the choice of a procedure (deterministic or heuristic) for construction/modification of solutions. In [JK16b, JK14b] the authors show that constructive steps within forward pass determine the main convergence properties of the BCOc. Most often, analysis of the performance of different underlying heuristics is not customary part of a tuning procedure, however, such scenario is not exclusive. To establish best set of heuristic rules we conduct independent series of experiments where we com-

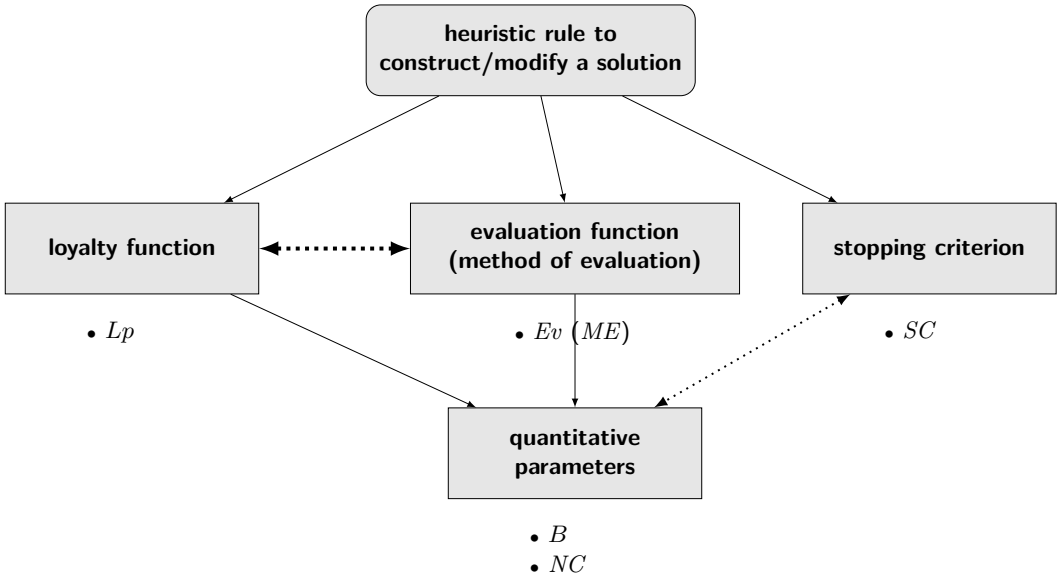


Figure 7.1: Hierarchy of parameters in the BCO algorithm design.

pare performances of several heuristic procedures to accompany development strategy of the BCO algorithm. After we determine the best constructive/modification rules, the evaluation function is considered as the most important among qualitative parameters with respect to design, as it reflects the objective of the problem.

The relation between BCO components is also illustrated in diagram 7.1. The position of a component is defined by its dependency from other parameters higher in the hierarchy. For example, selection of evaluation function depends on the choice of heuristic rules, however, it is independent from quantitative parameters B and NC. For the majority of the BCO implementations loyalty function would stand alone in the hierarchy diagram (or would not appear at all) as only a particular formula for probability decision (depicted by $p^{0,u}$, pg. 66) is utilized. To occasionally simplify notation while referring to the collection of loyalty functions, we introduce qualitative parameter Lp. Similarly, parameter Ev represents different possibilities of evaluation function, and ME indicates different methods of evaluations. New parameters Lp, Ev and ME take categorical values from the sets $\mathcal{L}$, $\mathcal{E}$ and $\mathcal{M}$, respectively. Another potential BCO parameter is stopping criterion, marked in Figure 7.1 as SC. This categorical parameter designates a type of stopping criterion, such as maximal number of iterations (N_{it}) or maximal CPU time (T). We do not utilize SC in our statistical tests and we draw our conclusion separately for each N_{it} and T. In particular, selection of the stopping criterion for a BCO algorithm is conducted carefully during the analysis of the best heuristic procedure and independently of other BCO parameters. It is worth noting that the experimental study of BCO conducted in this thesis imposes new questions concerning interactions between the loyalty and the evaluation function parameters, indicated by dotted line in Fig. 7.1. There are indications that the choice of loyalty function might depend on the values of the parameter Ev or ME. The most common interconnection is between numerical value of the stopping criterion and BCO quantitative parameters, B and NC. The interconnection originates from restricting maximal number of function evaluations or maximal CPU time. Both issues are to some extent addressed in this

dissertation.

The BCO parameters may be further classified on the basis of their dependency on the structure of a problem instance, thus, we differentiate: *problem-independent* and *problem-dependent*. Problem-independent components are: stopping criterion, size of population B, control number of forward/backward passes NC and loyalty paramater Lp. The second category includes parameters such as Ev or ME and the choice of heuristic rules in the forward pass. The border between the two categories is, occasionally, not obvious as it is usual to inspect each of the BCO parameters before recommendation, leaving an impression that each parameter is problem specific. We may find that strict limitations are imposed on the parameter NC, especially in the case of the constructive BCO algorithm. Therefore, it is important to understand the properties of each of the BCO's components that contribute to the robustness and flexibility while dealing with various optimization problems.

Stochastic nature of the BCO algorithm is induced by random number generator, defined by parameter $seed$. It represents a nuisance factor, however, is not included into the list of BCO parameters since it is independent from all others and is not problem specific.

7.3.5 Experimental setup for BCO analysis

Here, we refer to important specifics of our experimental study. The same computer system was used for empirical analysis of the BCO and its description is indicated bellow. In the study $seed$ is controlled by associating its value with an index of a run within an experiment.

7.3.5.1 Controlling experimental evaluation

Experiments are conducted by engaging one of the following stopping criterion: maximal number of iterations (N_{it}), maximal number of transformations ($MAXFLIP$) or maximal CPU time (T). Specifically for BCOc we employ N_{it} and T to investigate the performance. The study of BCOi is conducted using $MAXFLIP$ or maximal CPU time T as a stopping criterion. A performance measure is different between the two variants of BCO, and it depends on the considered optimization problem. Stopping criteria employed in the study of BCOc has enabled utilization of measures of central tendencies of data samples and helped provide reliable answers to our research questions. In particular, the study incorporates average value of solutions' qualities and/or their relative errors w.r.t. the known optimal solutions. In the study of BCOi a total number of transformations (*flips*) necessary to solve a given problem instance (i.e., until 3-CNF formula is satisfied) or to reach a stopping criteria is used. Furthermore, we utilize measures of central tendencies, generated over the set of 3-SAT problem instances (i.e., the average response values of each experiment is averaged over all problem instances).

7.3.5.2 Experimental environment

All experiments were conducted on Blade cluster with processors Intel(R) Xeon(R) CPU E5649 @ 2.53GHz and 24GB RAM, under Red Hat Enterprise Linux Server release 6.4 (Santiago) operating system, Kernel 2.6.32-358.11.1.el6.x86_64, gcc version 4.4.7. Implementation was done in C programming language. Compiler used is $g++$, with

optimization flag $O3$. It was noticed that the $O1$ flag generates slower executable compared to $O2$, and that $O3$ was actually the fastest. Statistical tests are implemented with R package language, version 3.2.5-1precise0 under Ubuntu 12.04.5 LTS operating system, Kernel 3.8.0-44-generic.

7.3.6 Final remarks

It is obvious that experimental tests provide a certain amount of freedom regrading the factors described in the first half of this chapter. A vast number of tuners for parametric stochastic algorithms are available in the recent literature. However, as stated at the beginning, the whole process from choosing the tuner to understanding how it works requires time. Some of the main issues with using the existing tuners is to identify if statistical tests are correctly implemented or if they consider all aspects of the appropriate design of experiments. For example, majority of the tuners are supported by non-parametric tests, not necessarily the choice we would make. The question of a correct choice of a statistical test in OR might originate from discrepancy of opinions found in the field of statistics. Namely, utilization of parametric tests is often disregard as it is considered to be too conventional. In addition, researchers tend to disregard checking for conditions that assist towards correct selection of a statistical test. Therefore, employing a tuner does not relieve us from the obligation to understand the requirements of the properer statistical analysis. According to [Bar95] experiments should use standard experimental techniques, designed to reduce variability of the measured outcomes by, e.g., producing more data points. Another recommendation is to use many instances of the corresponding problem class and provide comprehensive report of the results of computational testing summarized by measures of central tendency and variability. Due to all these issues, the application of automatic tuners remains the subject of future work.

We want to address here an issue of performance measure. A correct performance measure and stopping criteria are the most important part of each study, therefore, should be handled with caution. In the literature we can find various suggestions of correct choice of performance measure. [McG96] describes different aspects of computational experimental study of algorithms' properties and offers some general guidelines for choosing a good performance measure. The author points out the relevance of using a measure that exhibits small variability and recommends variance reduction techniques to provide reliable results. If data samples exhibit an unusual or bimodal distribution, it is recommended to observe the complete data (and not just estimators such as sample means). [Bar95] distinguish three categories of performance measure: solution quality, computational effort and robustness. Concerning the solution quality, the authors suggest that obtained solutions should always be compared to an optimal or a lower (upper) bound. They point out that best solution must be clearly identified as such.

In view of all this, we decided to use our own methodology and employ statistical tests and visual analysis that are standard in the field of statistics. We divide our study into small steps to contribute to reproducibility of our empirical conclusions. We justify utilization of each invoked statistical test and use large number of instances of the corresponding benchmark set. We summarize our results by either mean and standard deviation, in case of BCOc, or the average number of transformations averaged over the

problem instances, in case of BCOi. Since many researches advocate reporting of the running time [Bar95, Cof00, Joh02b], it is included in our resulting data. We provide graphical representations of the results as often as possible.

7.4 Chapter summary

In this chapter we emphasized the importance of conducting a thorough empirical analysis of a meta-heuristic algorithm. The chapter consists of three parts. In the first part we provide motivation of empirical study of a meta-heuristic method. We emphasize the importance of conducting systematic investigation in order to understand and predict behavior of the considered algorithm. In the second part we provide methodology of the experiments, gathered throughout the literature. In the third part we introduce *hierarchy diagram* of the BCO algorithm as an initial step to describe possible parameter interconnections. The diagram is also used to establish a course of empirical study.

Moreover, we provided:

- A short survey of experimental techniques for analyzing performance of meta-heuristics.
- Different interpretations of the measure of performance.
- Typical representatives of statistical methods when comparing performance of different (non)deterministic algorithms.
- Methodology behind the study of the BCO algorithms.

The empirical analysis represents important part of any algorithm's design, and therefore, in the next two chapters we concentrate on this topic. We conduct statistical and visual analysis of the BCO algorithm implemented for dealing with two different combinatorial problems, i.e., problem of scheduling and 3-SAT. Unexpectedly, conducting experiments and analyses of gathered results required a large amount of work in this dissertation.

CHAPTER 8

Development and empirical analysis of BCOc

The chapter is dedicated to the development, implementation and empirical analysis of the BCOc algorithm and, as such, is soliciting questions of design and tuning of BCOc. Inspired by work presented in Chapter 6, where the impact of quantitative parameters on the speedup is demonstrated, we conduct experimental analysis of the BCOc algorithm performance. In particular, we investigate an influence of method's parameters along different development stages of the algorithm. For the purpose of this chapter the empirical study is conducted on a set of randomly generated instances of $P||C_{max}$ (see Chapter 4). The results show that each of BCOc parameters can affect the algorithm's performance in different ways, leading towards outcomes that are *a priori* not known or are hard to predict for the given problem set. We demonstrate that the BCOc method exhibits good performance with respect to the solution quality and reliability.

The chapter is organized as follows. We first state the research goals of our study. Then, we investigate the structure of problem instances by graphical representation of their processing times distributions. Any information about the structure is not included in the BCOc algorithm, as it is commonly not available in real life. Before the comprehensive study of the BCOc algorithm, we conduct a comparative analysis between four heuristic algorithms. The selection and an experimental analysis of the best heuristic represents an important stage of the BCOc development. It helps to establish minimal value for the stopping criterion that avoids ceiling and floor effects. The study of BCOc proposes new evaluation function for $P||C_{max}$ to achieve good alternatives for reported solutions. Consequently, the empirical analysis of BCOc is extended from the basic parameters (B and NC) to the algorithmic components, i.e., loyalty functions and methods of evaluation. We follow two general guidelines during analysis of the results, *statistical* and *visual*, supported by an off-line algorithm configuration strategy. The strategy implies systematic adjustments of the BCOc parameters before the execution starts. Lastly, we use the opportunity to address important questions regarding the dynamics of the recruitment process within the backward pass.

8.1 Sensitivity analysis of the BCOc algorithm

Empirical study of BCOc considers a *static scheduling of independent tasks on identical machines*, $P||C_{max}$ (see section 2.3.2, pg. 29). BCOc is tested on the same benchmark

set as in [Dav12], thus, providing information that we can compare to. The main objective of this thesis is to capture features of the BCOc algorithm that lead towards general knowledge of its performance. We are motivated to prove that BCOc can produce better results if carefully tuned. Therefore, we conduct sensitivity analysis and compare all the loyalty functions from the literature and propose new evaluation function to examine if further improvements might be achieved. Sensitivity analysis consists of series of tests to detect behavior of the algorithm under different conditions. The conditions are related to problem structure, parameter values and the stopping criterion.

8.1.1 Research goals

Designed to assist the course of the research, the hierarchy diagram (Fig. 7.1, pg. 136) demonstrates the relations between the BCOc algorithmic components. We postulate that the optimal performance of BCOc depends on *a priori* selection of its qualitative parameters. The central point of the study is, therefore, founded in search for the best structural configuration of the BCOc parameters. However, as pointed in diagram 7.1, the qualitative factors L_p and ME might exhibit interactions. The question is to certain degree addressed in this chapter as we compare the results reported by various BCOcs defined by three methods of evaluation and ten different loyalty functions operating on configuration sub-space $\mathcal{B} \times \mathcal{NC}$. Along the main study, the experiments reveal influence of different problem instances and expose significant interactions between quantitative BCO parameters. The purpose of this chapter is summarized in the list of research goals.

- Q_1 Identify the most influential BCOc parameter.
- Q_2 Compare new evaluation function ev_2[(1)] against ev_1.
- Q_3 Investigate robustness of BCOc parameters to structure of problem instances.
- Q_4 Identify the most successful loyalty functions and the reasons behind their success.
- Q_5 Examine differences between results within two stopping criteria: maximal number of iterations and maximal allowed time.
- Q_6 Does the selected experiment provide information about the BCO performance in general: can results of the experiment, obtained for case N_{it}, predict the performance of the algorithm for longer runs?
- Q_7 Can we determine the maximal number of the bees? The question relates to cases where the improvement in solution quality is noticed with an increase in the number of bees. Therefore, we ask if the study helps to determine if there exists a maximal number of bees after which the quality deteriorates.

Collected answers to the above research questions might help to improve the design of the BCOc algorithm for other similar combinatorial problems.

[(1)] To distinguish from the notation that is commonly used for objective function, an evaluation function is denoted as ev.

8.1.2 Test instances

Instances used to test BCOs and candidate heuristics are introduced in [Dav06b] for MSPCD (see Section 2.3.1.1, pg. 27). They represent randomly generated instances defined w.r.t. three different factors: number of tasks, task graph density, and number of identical machines. The problem instances are specified as Iogra$< n >_< \rho >_< m >$, where n designates number of tasks, ρ the graph density and m denotes number of machines. Because the $P||C_{max}$ problem does not deal with task dependencies, the graph density is not important for presented work (is set to zero). In addition, for independent tasks the connection between machines is irrelevant.

The structure of a problem instance can be investigated by observing distributions of its tasks computational times. Bar charts of task processing times (task lengths) for each m and n are provided in Figs. B.8–B.16 (Appendix B, pg. 306). By observing bar charts we may conclude that the occurrences of tasks' processing times evidently exhibit extremely skewed (spread) distributions and do not follow any known distribution. Therefore, we employ descriptive statistical tools, i.e., box-and-whiskers plot [Tuk77] (see section A.1.2, pg. 247). Box-plots are suitable for comparisons between different groups of data with skewed distributions. Fig. 8.1 shows box-plots for two classes of instances ($m = 12$ and $m = 16$) and different problem-size (ranging from 100 to 500).

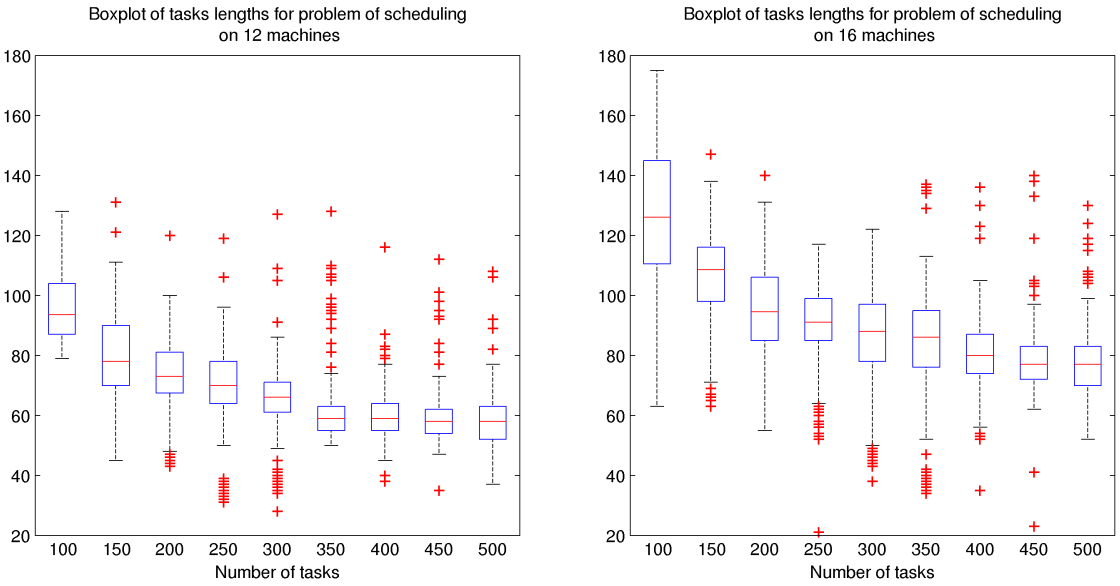


Figure 8.1: Box-plot for $m = 12, 16$ and 9 instances in relation to the n, where possible outliers are marked with red crosses.

The box-plots in Fig. 8.1 indicate that instances for $m = 16$ have longer computing times. This implies that class $m = 16$ is more versatile, i.e., tasks lengths range from 20 to 180 time units, whereas range of tasks' lengths in class $m = 12$ goes from 30 to 130. Another observation is that distribution of tasks' lengths for $n = 100$ does not show outliers. Namely, the processing times are nicely grouped compared to distributions of tasks' lengths when $n > 100$. Comparing between instances shows while n is growing

the number of shorter tasks is increasing at the expense of longer. In Fig. 8.1 this is indicated by the decrease in median of the tasks' lengths. According to [Dav06b] the property is deliberate.

8.1.3 The first BCOc for $P||C_{max}$

The first implementation of the BCOc for $P||C_{max}$ is presented in [Dav09]. The authors use the stochastic version of the LPT rule for selecting tasks (Section 2.3.2, pg.30): tasks with a longer processing time have a higher chance to be chosen. The machine selection strategy is based on concepts of ES, where machines with shorter execution times have higher probability to be selected. In [Dav12] the machine selection strategy has been inspired by BPP and authors incorporate *best-fit* heuristic into the BCOc model. Because implementation of the heuristic requires setting the capacity of the machines loads (C), the authors suggest an adaptive strategy. Namely, at the beginning of an iteration variable C is equal to the theoretical lower bound of tasks' computing times. After the new best solution is found, C takes the value of its makespan. We elaborate on different settings of variable C later in this chapter.

It is worth noting that, to obtain fair comparison among stochastic and a deterministic algorithm on the considered benchmark set, an order in which tasks are picked and in which they are scheduled should not, respectively, exploit original task enumeration and the deterministic ES strategy. The reason arises from the way *Iogra* instances are reported in [Dav09]. If tasks are selected in order of their appearance in the input file and are scheduled by ES rule, the optimal solution is obtained as a first feasible solution in matters of microseconds on any new-age CPU architecture. Therefore, their order should be rearranged before any run.

8.2 Candidate heuristics for $P||C_{max}$

Development of meta-heuristics methods for the particular optimization problem requires selection of a heuristic rule specifically designed to tackle the problem. For $P||C_{max}$ we review/develop several stochastic greedy procedures for selection of tasks to be scheduled and the machine on which the tasks should be allocated. The results of these rules are denoted as *task-machine pairs*. Selection of a task-machine pair can be conducted in numerous ways, yielding various heuristic algorithms for $P||C_{max}$. We consider algorithms implemented as stochastic iterative heuristic procedures that incorporate greedy rules for task-machine selection. We compare four different heuristic procedures to determine the best one. The choice of heuristics is based on reports about their success from the literature. The algorithms generate a complete solution at the end of each iteration and stop upon reaching a maximal number of iterations. The algorithms are designed to be robust to changes in the structure of the problem. At the beginning of a single iteration most or all decisions made in previous one are forgotten, failing to systematically explore the search space.

Our first scheduling procedure is the adaptation of the well known greedy method, the LS algorithm, which incorporates the LPT rule. New scheduling procedure is denoted as `sLPT+ES` and consists of two parts: stochastic LPT (`sLPT`) and `ES` strategy. The `sLPT` procedure deals with the task selection and is implemented in a way that tasks

with longer processing time have a higher probability to be picked from the list L. In the second step, sLPT+ES allocates the task to the least loaded machine. The second procedure we investigate replaces the deterministic scheduling with the stochastic, basing both decision rules on probabilities. The new procedure is denoted as sLPT+sES where the scheduler utilizes a stochastic earliest start strategy (sES): the probability to schedule on a machine with minimal load is the largest. The last two heuristic procedures, sLPT+FF and sLPT+BF, are inspired by the techniques employed for BPP. Essentially, they consist of two steps: i) tasks are sorted following sLPT rule, and ii) each task is assigned to a machine by utilizing either *first-fit* or *best-fit* technique. All the corresponding heuristic algorithms are denoted in the same manner, and their description (including pseudo-codes and the solution representation) is presented in Appendix A (pg. 248).

8.2.1 Experimental evaluation of candidate heuristics

Here, we compare four heuristic algorithms with an objective to select the best candidate for the BCOc method. The experimental study was conducted on 12 different problem instances, following the recommendations from Section 7.3.5 (pg. 137): an experiment consists of 100 independent runs of a particular algorithm with the controlled value of *seed*. Two instances are chosen ($n = \{100, 250\}$) from each class $m = \{2, 4, 6, 8, 12, 16\}$. The corresponding results are presented in Table 8.1. The first column identifies the type of a problem instance. The second, sixth, tenth and fourteenth column provide the average solution's quality ($\bar{y}$) and the corresponding standard deviations (*s.d.*) in the row bellow. Columns $\bar{t}$ show average times of the "first hit" and the corresponding *s.d.*. Time of the first hit represents the first time the best solution was obtained during a single run of the algorithm. All running times are measured in milliseconds. Columns $\overline{n_{it}}$ show the average number of iterations to obtain the best solution. The quality of the best solution found during one experiment is reported in the last column.

Table 8.1 shows significant differences between the average quality of solutions reported by the heuristics. If the number of machines is fixed to a small number, all test instances are easily solved either to optimality (for $m = 2$) or very close to optimal (for $m = \{4, 6, 8\}$). The same conclusion is reported in [Dav12] for both CPLEX and BCOc. By comparing columns $\bar{y}$ we conclude several results. For $m = 12$ and $n = 100$, sLPT+ES generates higher quality solutions compared against the others. However, for $n = 250$, the algorithm is worse than the *fit*-heuristics, which have performed significantly better. For class $m = 16$ the sLPT+ES algorithm generates the best results w.r.t. $\bar{y}$. sLPT+sES can be considered as the worst heuristic in this study. Finally, when we consider the *best found* solution, in majority of the cases the *fit*-heuristics outperform ES-strategies.

The reason why solution quality for $m \geq 12$ degrades for *fit*-heuristics might be due to a "slow start". Namely, *best-fit* and *first-fit* schedulers require more iterations than the *ES*-strategies to obtain the best value for the capacity (Fig. A.1, pg. 249). The *ES*-strategies show stagnation in the solution quality after the initial improvement phase. In particular, the sLPT+sES algorithm exhibits the worst performance on all the problem instances w.r.t. the best found solution and the average performance, thus is eliminated from the further analysis.

To compare the remaining three heuristic algorithms sLPT+BF, sLPT+FF and sLPT+ES

Table 8.1: Scheduling results for test problems with known optimal solutions appropriated from [Dav06b] with $n = \{100, 250\}$, $n_{run} = 100$, $N_{it} = 100$ and different number of machines.

Problem	sLPT+BF				sLPT+FF				sLPT+ES				sLPT+sES			
	$\bar{y}$ (s.d.)	$\bar{t}$ (s.d.) $[10^{-3}]$	$\overline{n_{it}}$ (s.d.)	best	$\bar{y}$ (s.d.)	$\bar{t}$ (s.d.) $[10^{-3}]$	$\overline{n_{it}}$ (s.d.)	best	$\bar{y}$ (s.d.)	$\bar{t}$ (s.d.) $[10^{-3}]$	$\overline{n_{it}}$ (s.d.)	best	$\bar{y}$ (s.d.)	$\bar{t}$ (s.d.) $[10^{-3}]$	$\overline{n_{it}}$ (s.d.)	best
Iogra100_2	800.00	0.86	6.26	800	800.21	0.47	7.68	800	800.00	0.96	13.52	800	800.00	1.08	15.22	800
	± 0.00	± 0.24	± 5.37		± 2.07	± 0.50	± 10.47		± 0.00	± 0.28	± 14.74		± 0.00	± 0.26	± 12.28	
4	800.91	1.36	41.52	800	800.95	0.74	41.04	800	802.51	1.33	41.72	801	803.83	1.82	55.57	801
	± 0.45	± 0.36	± 25.93		± 0.38	± 0.54	± 23.55		± 0.92	± 0.37	± 27.09		± 1.36	± 0.53	± 28.34	
6	803.04	1.66	59.70	801	803.10	1.03	56.83	801	805.36	1.28	37.88	802	810.60	1.85	56.43	805
	± 0.80	± 0.40	± 25.10		± 0.77	± 0.65	± 24.24		± 1.24	± 0.35	± 25.04		± 2.85	± 0.58	± 28.93	
8	805.11	1.94	72.92	803	805.25	1.23	68.62	801	811.57	1.42	44.72	807	824.25	1.97	59.83	809
	± 1.13	± 0.33	± 18.04		± 1.21	± 0.60	± 18.81		± 1.97	± 0.47	± 30.42		± 5.30	± 0.61	± 28.00	
12	833.75	2.51	94.87	818	832.85	1.80	93.83	819	826.93	1.58	49.79	816	853.75	1.78	45.33	831
	± 22.30	± 0.25	± 7.72		± 18.83	± 0.57	± 8.58		± 3.22	± 0.48	± 27.36		± 7.79	± 0.74	± 33.28	
16	903.47	2.79	99.07	811	899.83	1.86	98.87	811	834.07	1.65	46.21	820	884.17	1.61	36.33	849
	± 67.58	± 0.21	± 3.74		± 67.20	± 0.51	± 3.14		± 4.04	± 0.54	± 29.67		± 14.20	± 0.75	± 33.78	
Iogra250_2	1400.00	1.35	4.98	1400	1400.00	0.49	5.11	1400	1400.00	1.57	9.44	1400	1400.00	1.84	11.71	1400
	± 0.00	± 0.31	± 4.20		± 0.00	± 0.54	± 3.83		± 0.00	± 0.52	± 10.28		± 0.00	± 0.70	± 11.62	
4	1400.84	3.34	44.78	1400	1400.85	1.73	42.09	1400	1401.74	3.10	41.22	1401	1402.64	3.99	46.13	1400
	± 0.46	± 1.32	± 26.27		± 0.52	± 1.15	± 25.72		± 0.72	± 1.49	± 31.21		± 1.25	± 1.72	± 27.97	
6	1402.24	4.14	56.92	1401	1402.21	2.59	61.11	1400	1404.58	3.09	39.40	1401	1407.89	4.56	53.21	1402
	± 0.67	± 1.05	± 19.51		± 0.71	± 1.11	± 20.83		± 1.00	± 1.47	± 28.55		± 2.01	± 1.72	± 26.67	
8	1404.75	5.30	73.70	1403	1404.73	3.23	72.68	1402	1407.98	3.33	41.73	1405	1417.12	4.60	51.49	1409
	± 0.91	± 1.02	± 17.28		± 1.04	± 1.16	± 17.93		± 1.51	± 1.46	± 27.15		± 3.86	± 2.00	± 30.19	
12	1408.46	6.63	88.03	1405	1408.39	3.80	86.18	1404	1412.40	3.70	45.22	1407	1435.53	4.47	47.54	1420
	± 4.33	± 0.73	± 11.27		± 3.38	± 1.04	± 11.47		± 1.97	± 1.68	± 29.50		± 7.06	± 2.15	± 31.02	
16	1494.73	7.84	99.35	1408	1474.32	4.07	98.06	1409	1424.38	3.85	44.86	1417	1460.48	3.52	32.35	1434
	± 56.54	± 0.28	± 2.62		± 57.67	± 0.75	± 5.52		± 2.99	± 1.77	± 29.15		± 9.64	± 2.45	± 33.40	

we conduct another study according to recommendations in [Yan13]. The experiment is conducted on one problem instance and each algorithm runs until a stopping criterion is met. We define the stopping criterion as the condition determined by a given value of *tolerance* δ, such that $|y_{gmin} - y^*| \leq \delta$, where y_{gmin} is the makespan of the best overall solution of a particular experiment and y^* is the makespan of the optimal solution. The number of iterations needed to satisfy the condition, $n_{it}(\delta)$, is reported at the end of each run. The experimental methodology is same as previously (see also pg. 137).

The comparison of the three heuristics is based on the identification of the minimum number of iterations needed to reach the target value for *Iogra100_12* as the representative of hard instances with known optimal value ($y^* = 800$). For further comparison with BCOc, the tolerance value is set to $\delta = 11$ and results are presented in Table 8.2. The tolerance value corresponds to the best solution reported for BCOc and $B = 5$ and $NC = 10$ [Dav12]. The table contains three sections, one for each reviewed heuristic algorithms. The first column provides an average value of the number of iterations to reach the target value ($\overline{n_{it}}(\delta)$) and the corresponding standard deviation in the row bellow. The second column shows an average time ($\bar{t}$) and the third column an average quality of the solutions generated during the experiment ($\bar{y}$). The average solution quality is used because, in the most cases, the algorithms have not reached the target value and, instead, have generated a better solution.

According to Table 8.2 `sLPT+ES` generate significantly different $\overline{n_{it}}(\delta)$ than the other two algorithms and has exhibited the worst performance. Both *fit*-heuristics produce better results and in average similar outcomes for the given problem instance, reaching the target solution in nearly the same time ($\approx 0.09s$). Compared to Table 8.1, reaching the target value requires more time and larger number of iterations in each run of the experiment. `sLPT+FF` requires larger number of iterations than `sLPT+BF`, indicated by vales in columns $\overline{n_{it}}(\delta)$. To decide between the two algorithms, we compare standard deviations in each column. `sLPT+BF` shows lower variability and therefore is used as an underlying constructive procedure in BCOc.

Table 8.2: Minimal number of iterations needed to obtain value 811 for test instance Iogra100_12 and $n_{run} = 100$. Time is reported in seconds.

`sLPT+BF`			`sLPT+FF`			`sLPT+ES`		
$\overline{n_{it}}(\delta)$	$\bar{t}(\delta)$	$\bar{y}(\delta)$	$\overline{n_{it}}(\delta)$	$\bar{t}(\delta)$	$\bar{y}(\delta)$	$\overline{n_{it}}(\delta)$	$\bar{t}(\delta)$	$\bar{y}(\delta)$
8001.66	0.090	810.86	8878.86	0.088	810.79	2806941.65	27.85	810.43
±5423.97	±0.060	±0.42	±5639.58	±0.055	±0.50	±3591269.91	±35.60	±0.91

8.2.2 Conclusions regarding the best heuristic

Results of previous sections motivated us to conduct an additional experimental study of `sLPT+BF`. In particular, we have been interested in the complexity of problem instances w.r.t. m and n. Preliminary results, reported in [Dav12] suggest that m increases the complexity of the problem. Extensive analysis of `sLP+BF` (Section A.1.5, pg. 253) indicate other conclusion. The second research question concerns the analysis of the stagnation phase, i.e., the minimal value of N_{it} after which the algorithm does

not improve the quality of the solution. Therefore, to avoid ceiling effects during the empirical analysis of BCOc we are concerned with producing the most suitable value of the maximal number of iterations.

The first result of `sLPT+BF` study in Section A.1.5 is that algorithm generates high quality results for all test instances and, therefore, is a robust and powerful algorithm for dealing with $P||C_{max}$. To avoid ceiling effects when comparing different BCOc instances the maximal number of iterations should be set to 100. Namely, an examination of the descriptive statistics in Tables A.1 and A.2 illustrate that the increase in the number of iterations does not have a significant impact on performance for problem instances $n \geq 150$. Influence of different values of N_{it} on the solution quality is also represented by graphics in Fig. A.5. The maximal value of iterations is 10000 and for the majority of considered problem instances the outcomes after a 200 iterations do not seem to be practically different.

8.3 Development of the BCOc algorithm for $P||C_{max}$

Based on the study from the previous section, we confirm that the selection of `sLPT+BF` procedure for constructing partial solutions within the forward pass in [Dav12] is appropriate. Therefore, we re-implement the BCOc algorithm with an adaptation that assures the acceptance of all the values from the extended domain of NC. We use the same benchmark set as in [Dav12]. The authors report the best configuration for quantitative parameters for loyalty function $p^{0,u}$, i.e., $B = 5$ and $NC = 10$. In this chapter we show that, by extending configuration space, the performance of BCOc could be improved within the same N_{it}. In section 8.2 we have elaborated why $N_{it} = 100$ is suitable for our empirical study. Moreover, we revealed that underlying heuristic of BCOc is powerful as a standalone procedure to procure high quality sub-optimal solutions. The objective is, therefore, to demonstrate that well tuned BCOc can produce higher-quality results. In addition to loyalty functions described in Section 4.2.3.1, we propose a new evaluation function (ev_2) and investigate its impact on the results of BCOc. Utilization of function ev_2 has suggested a new line of study. In particular, inducing the maximization strategy of an evaluation function might guide the search towards the high-quality sub-optima. The evaluation strategies are defined as the new BCOc parameter ME, introduced previously in Section 7.3 (pg. 134). Therefore, influence of evaluation function is studied by exploring different values of parameter ME.

Regarding the stopping criteria we propose two independent studies. The experiments are developed under the maximal number of iterations (N_{it}) and maximal allowed CPU time (T). Motivation to impose maximal allowed CPU time proceeds from the literature. Namely, it is common to restrict the maximal *number of evaluations* (Nev) while comparing two meta-heuristic methods. Parameter Nev estimates the computational effort independently from the computer architecture and its environment. However, imposing its values disregards values for one of the BCOc quantitative parameters. Therefore, to stay within the common framework of the BCOc design, the closest to achieve the same results is to employ T. It is worth noting that algorithm instances might execute several iterations more or less then the others until stopping criterion T is satisfied. The discrepancy is a result of time functions utilization (`getrusage` and `gettimeofday`) influenced by background processes of the computer operating system.

In addition to parameter/structural tuning, the central point of our empirical study is to learn about the mechanisms that guide the BCOc algorithm towards the high quality solutions.

8.3.1 Experimental methodology

8.3.1.1 Solution representation

Following the conventions from [Dav12], the solution is represented as a (dynamic) matrix $S_{m\times n}$, where the element s_{ji} represents the index of the i-th task scheduled to the j-th processor. The list of all task lengths (processing time of each task) is kept in the vector $L = (l_1, \ldots, l_n)$. After allocating $k < n$ tasks, the auxiliary list of lengths of $n - k$ non-scheduled tasks is saved in the vector L'. Concerning the processor loads, two vectors are used: $O = (o_1, \ldots, o_m)$ where element o_j denotes the number of tasks allocated to j-th processor and $Y = (y_1, \ldots, y_m)$ with elements y_j that represent the sum of processing times of all tasks allocated to processor j.

A solution of each bee $b \in 1, \ldots, B$ is, at the end of iteration, saved in the data structure $(S_b, O_b, Y_b, y_{bmax})$, where y_{bmax} indicates makespan of b-th bee solution. At the end of each iteration, all solutions are compared and the best-so-far saved within a global structure g_{best} consisting of $(S_{gmin}, O_{gmin}, Y_{gmin}, y_{gmin})$. Furthermore, data of the best found solution x_{best} may be presented as $(S_{gmin}, O_{gmin}, Y_{gmin})$, while y_{gmin} refers to its objective value. Therefore, the result generated by the BCOc instance, corresponds to the evaluation value of the best solution, i.e., to its makespan (y_{gmin}).

8.3.1.2 Performance measure

The study of BCOc is founded on the solution quality. Other measures, such as success rate, could also be used, however, only when knowing what constitutes an acceptable target result and is, therefore, disregarded. To compare BCOc instances, we use statistics of central tendency of the acquired samples of data. Because highly discrete nature of the response values, median of each experiment does not help decide on the best results. Therefore, mean value, together with its standard deviation, is used to decide about configurations of parameters that maximize algorithm's performance. Moreover, to evaluate improvements, the BCOc algorithm is compared against the underlying heuristic `sLPT+BF`. Because optimal solutions are known, we frequently employ absolute and relative errors[(2)] to generate graphics.

8.3.1.3 Stopping criteria

Specifying stopping criterion belongs to the type of BCO's components that produces different outcomes in performance. Therefore, we establish two independent studies w.r.t: (1) maximal number of iterations; (2) maximal allowed CPU time.

The first study implies that $N_{it} = 100$ for several reasons. Firstly, to initiate a fair comparison between BCOc instances we avoid potential ceiling effects. Secondly, results in Fig. A.5 and Table A.1 demonstrate that the heuristic algorithm `sLPT+BF` requires at least 100 iterations until it starts to generate high quality solutions. In

[(2)] Relative errors are mostly reported as *percent errors*, i.e., $p.err = r.err \times 100$. In order to avoid ambiguity with the *percentage errors* of the standard deviation, instead *percentage relative error* is kept.

particular, we observe that after 200 iterations the algorithm's performance reaches a ceiling effect. Employing large number of bees induces an increase in the number of evaluations, thus, yields a ceiling effect earlier then 200 iterations. Finally, restricting execution with N_{it} helps minimize an influence of background system process that are running concurrently with our tests. In addition, we can explore algorithm's performance when a small amount of computational cost is provided. It is worth noting that the BCOc algorithm requires different running times for quantitative parameter configurations. For example, average running time for 20 bees approximately doubles compared to $B = 10$. For NC the predictions are not as clear. Nevertheless, a general trend exists as time to perform NC steps increases along the values of the parameter.

Values of N_{it} may be used to estimate N_{ev}, which we demonstrate on a simple case. Let parameters $B = 10$, $NC = 15$ and $N_{it} = 100$. Then, the evaluation function is called $B * (NC - 1) * N_{it} = 10 * 14 * 100 = 14000$ times. Moreover, limiting the number of evaluations (operations counts) requires reporting an average time per count due to stochastic nature of BCOc, which is another reason why we do not employ Nev.

In case of maximal allowed CPU time, values for T are determined as a function of the problem size (n), i.e., $T = n/100$ $[s]$. Compared to $N_{it} = 100$, for each n values of T coincide with the estimated running times of BCOc instances with the maximal values for quantitative parameters. Therefore, in the second independent study almost all BCOc instances acquire larger number of iterations ($N_{it} \geq 100$), which might be used for empirical study of BCO's convergence properties. Furthermore, setting T procures unbiased comparison in terms of computational effort between different BCOcs w.r.t. B and NC.

8.3.1.4 Quantitative parameters

To collect necessary data, we cover large part of the configuration space. Elaboration for qualitative parameters is presented in the following Section 8.3.3, while here we focus on the quantitative parameters.

The values of quantitative parameters are restricted either subjectively or influenced by greedy rules. To encompass parameter values from the literature the domain of parameter B is set to the interval $[1, 20]$. It is large enough to enable identifying an increase/decrease in the solution quality. Values for the parameter NC are not carefully elaborated in [Dav12], therefore, we define its domain with regard to limitations imposed by the dimension of the problem instance and the acceptable computational time. Namely, problem instances with n tasks impose study of domain $[1, n]$ of parameter NC. As a result, exploration of the domain for $n \geq 150$ requires large amount of computational resources. Therefore, our study is confined to $NC \in [1, 100]$. This restriction is best suited for test instance $n = 100$, as it covers the complete configuration space.

8.3.2 Design of BCOc forward pass

The BCOc algorithm for $P||C_{max}$ generates solutions by invoking rules of the `sLPT+BF` heuristics. Iteration of the BCOc begins with an empty solution assigned to each bee. The first task-machine pair is determined randomly. The rest of the task-machines pairs are established following the `sLPT+BF` rule. The number of constructive moves in BCOc

is determined as a function of parameter NC. Specifically, for $P||C_{max}$ approximately n/NC components are added to the current partial solution during each forward pass. Procedure that describes the implementation of this calculation is presented in Appendix A.2.1 (Fig. 1, pg. 258). In fact, the description of the BCOc algorithm is closely related to specifications of the `sLPT+BF` algorithm (see pseudo-code A.1, pg. 249).

8.3.3 Design of BCOc backward pass: qualitative parameters

Backward pass implies evaluation of partial solutions with formula (4.1) (Chapter 4, pg. 63). In particular, probabilities to remain loyal to a found (partial) solution are determined by one of the expressions in Sections 4.2.3.2–4.2.3.11. After resolving on loyalty, bees enter the recruiting process as the last phase of the backward pass. The recruiters are those bees that remain loyal to its solution. As mentioned earlier, the remaining (uncommitted) bees are followers that must determine which recruiter's partial solution they will continue to construct (Sections 4.2.2.1). Commonly, to make this decision we use roulette wheel and formula (4.5)(pg. 65). The chosen solution, described by the structure $(S_b, O_b, Y_b, y_{bmax})$, is copied from the recruiter to the follower.

Important segment of our study is the structural tuning, i.e., influence of qualitative factors Lp and ME (Ev) on the measured outcomes. Lp is the method specific and ME is the problem specific parameter. All loyalty functions collected throughout the literature are also considered: $p^{0,u}$, p^1, p^2, $p^{3,n_{iter}}$, $p^{4,u}$, $p^{5,u}$, $p^{6,u}$, $p^{7,u}$, p_b^8 and $p^{9,u}$.

Unlike the standalone heuristic algorithm, where only the complete solutions are assessed, in BCOc we also need to evaluate partial solutions. In this study, we considered two evaluation functions:

$$(1)\ ev_1 = y_{bmax} \quad (2)\ ev_2 = \frac{y_{bmax}}{S'_b}.$$

Evaluation function ev_1, introduced in [Dav12], relies on the value of *makespan* of each (partial) solution (y_{bmax}) and is, therefore, more receptive due to lower computational costs related to its utilization. Evaluation function ev_2 depends on two parameters: y_{bmax} and S'. Function ev_2 associates the better quality to partial solutions with larger values of the function. Analysis of two evaluation functions initiates introduction of the parameter ME (pg. 134).

Consequently, we distinguish four different methods of evaluations, among which three are investigated, as it is elaborated in the remainder of this section. In case of ev_1, both maximization and minimization principles are used, therefore, producing two methods of evaluations. Maximization principle evaluates a partial solution with the largest *makespan* (the worst case w.r.t. the objective of the problem) as the best among B partial solutions, i.e., the normalized value of the function ev_1 equals 1. Normalized value of ev_1 of the partial solution with the lowest *makespan* is then appointed to 0. This method of evaluation is denoted as $\max(ev_1)$. In the case of minimization principle, the lowest *makespan* among B partial solutions, (i.e., the best case w.r.t. the objective of the problem), is marked as the best and the corresponding value of ev_1 is normalized to 1. The value of ev_1 of the partial solution with the largest *makespan* is normalized to 0. This method of evaluation is denoted as $\min(ev_1)$. Justification to incorporate ME can be explained by characteristic of the underlying heuristic `sLPT+BF`.

Evaluation function ev_2 is introduced to determine whether performance of BCOc can be improved by incorporating knowledge about the computational time of non-scheduled tasks. Namely, a partial solution that already contains longer tasks is set as the better one. We follow the reasoning of LPT heuristics stating that it is easier to schedule shorter tasks within current partial solution. We quantify this behavior as the ratio between the current *makespan* of the solution an the corresponding sum of the non-scheduled tasks S'_b. Incorporation of $max(ev_2)$ might help to generate better solutions compared against $min(ev_1)$. To test this hypothesis, in the following text we analyze different scenarios during a scheduling process.

Let us observe two partial solutions ($B = 2$) and let S'_b, $b = 1, 2$, be corresponding sums of the processing times of non-scheduled tasks. We distinguish two scenarios: (1) corresponding partial solutions have the same makespan $y_{bmax} = y_{max}$ and different S'_b; (2) partial solutions have the same sum of non-scheduled tasks $S'_b = S'$. We observe evaluation function ev_2 as a function of two parameters, defined for each bee as $ev_2 : \mathbb{Z} \times \mathbb{Z} \to \mathbb{R}$ for a problem of minimization of function f.

Scenario (1) describes the case of two bees with equal y_{bmax} and different sums of non-scheduled tasks lengths. Let partial solution of Bee_1 contain shorter tasks relative to Bee_2. Therefore, the sum of non-scheduled tasks of Bee_1 is larger than of Bee_2, i.e., $S'_1 \geq S'_2$. It holds that

$$S'_1 \geq S'_2 \iff \frac{1}{S'_1} \leq \frac{1}{S'_2} \iff \frac{y_{max}}{S'_1} \leq \frac{y_{max}}{S'_2} \iff ev_2(\text{Bee}_1) \leq ev_2(\text{Bee}_2),$$

i.e. , the value of ev_2 of Bee_2 (that has to schedule tasks with shorter sum S'_b) is greater than of Bee_1. Obviously, we want that Bee_2 achieves higher probability to remain loyal. In particular, we allow partial solutions with smaller sum of non-scheduled tasks to propagate within the population since it is easier to allocate shorter tasks after the allocation of longer ones. However, we are not taking into account the distribution of the tasks lengths, which might influence results in practice. In scenario (2) sums of non-scheduled tasks lengths S'_b are the same for each bee. Accordingly, sums of scheduled tasks of each bee are also equal, while y_{bmax} might differ. With regard to the objective function, the previous discussion about function ev_1 and the underlying heuristic, both maximization and minimization of evaluation function may be employed.

By a group of preliminary experiments we conclude that $\max(ev_2)$ strategy yields better results than $\min(ev_2)$. Therefore, we define set of methods of evaluation to be $\mathcal{M} = \{min(ev_1), max(ev_1), max(ev_2)\}$.

8.3.4 Collection of results

The region of interest, on which we conduct the empirical study of BCOc for $P||C_{max}$, has been established in the previous section and all the values of parameter configurations are summarized in Table 8.3.

The subject of this section is to describe the first step of the empirical study of BCOc: the collection of results. Namely, a product of each experiment is a set of data. The data set contains a collection of measured outcomes (response values) generated by the corresponding BCOc instance. At the end of a single run the BCOc instance reports three different values:

y – the evaluation value of the best-found solution x_{best},

t – time to obtain x_{best} for the first time,

n_{it} – number of iterations needed to obtain x_{best} for the first time.

Moreover, the BCOc algorithm reports complement of the stopping criterion, i.e., maximal CPU time (T) if N_{it} is the stopping criterion, and total number of iterations (N_{it}) if T is the stopping criterion. The response variables are summarized in Table 8.4.

Table 8.3: Parameter space for experimental analysis of BCOc.

Parameter	**Domain**
ME	$\{\min(ev_1), \max(ev_1), \max(ev_2)\}$
loyalty function	$\{p^{\alpha,\beta} : \alpha = \overline{1,20}\}$
B	$\{1,\ldots,20\}$
NC	$\{1,\ldots,100\}$
n_{run}	$\{100\}$
seed	$\{1,\ldots,n_{run}\}$

Table 8.4: Response values after a run of the BCOc instance.

Stopping criterion	Best solution y	Min. time t	Min. iter. n_{it}	Total time T	Total iter. N_{it}
N_{it}	✓	✓	✓	✓	
T	✓	✓	✓		✓

The measured outcomes of the experiments are employed in computing descriptive statistics, i.e., mean, range, minimal value, maximal value, coefficient of variation and the corresponding standard deviations. In statistics, these values represent central tendencies and variation for each unique configuration of the BCOc parameters and are common in operations research community. The total number of statistics of one experiment amounts to 24 different values. Additionally, the total number of experiments is large as for the single instance we need to perform $|\mathcal{M}| \cdot |\mathcal{L}| \cdot |\mathcal{B}| \cdot |\mathcal{NC}| = 3 \cdot 10 \cdot 20 \cdot 100 = 60000$ experiments. The focus of the BCOc study is on the quality of a solution, therefore, we mostly utilize average response value $\bar{y}$. Nevertheless, we document other characteristics of the BCOc performance which we address during the course of the study. We concentrate on four average outcomes and their corresponding variability, as shown in Table 8.5.

The average response values of the experiments are reported in Tables 8.6–8.9 and Tables B.2–B.33, organized in the following fashion. Each table corresponds to one problem instance and consists of three groups of data, distinguished by method of evaluation. Columns are arranged to show descriptive statistics of the best and the worst experiment. The first column indicates the loyalty function. The second column indicates the quality of the data, i.e., results of the best (b) and the worst (w) experiment.

Table 8.5: Overview of the response values and the corresponding descriptive statistics of the experiment.

Stop. crit.	Mean of y $\bar{y}$	Range of y Δy	Mean n_{it} $\pm$ st. dev. $\overline{n_{it}} \pm s$	Mean $t \pm$ st. dev. $\bar{t} \pm s$	Mean $T \pm$ st. dev. $\bar{T} \pm s$	Mean N_{it} $\pm$ st. dev $\overline{N_{it}} \pm s$
N_{it}	✓	✓	✓	✓	✓	
T	✓	✓	✓	✓		✓

Remaining three groups of seven columns provide statistics for the most successful (upper row) and the worst (bottom row) among $B \cdot NC$ experiments conducted for the same combination of qualitative parameters. In particular, column $\bar{y}$ shows the best (the worst) mean value for solution quality, followed by column Δy containing the range, i.e., difference between minimal and maximal value of the data sample for y. To describe variation in $\bar{y}$ we opted for range instead of standard deviation to emphasize the span of solutions' quality under the influence of $seed$. Columns B and NC show configurations of the corresponding BCOc parameters that generate the best (worst) $\bar{y}$. Additionally, provided are: average value of computational effort such as number of iterations and running time until generating the best solution for the first time, mean of required CPU time (if stopping criterion is N_{it}) or mean of maximal number of iterations (if stopping criterion is T). The reported values of CPU time are expressed in miliseconds.

The best experiment according to reported values in the Tables 8.6–8.9 and Tables B.2–B.33, is considered to be the one that, on average, generated the lowest value. For each value of ME the best $\bar{y}$ is colored in orange in such a way that the most intensive colored field signifies the overall best. The overall best is also marked with the asterisks (*). The average outcome $\bar{y}$ might be hard to identify outside the specified tables, and its suitable indexation is as follows.

$$\bar{y}_{i,j,k,l} = \frac{1}{n_{run}} \sum_{s=1}^{n_{run}} y_s \text{ where } i \in \mathcal{M}, j \in \mathcal{L}, k \in \mathcal{B}, l \in \mathcal{NC}, s = seed. \tag{8.1}$$

where y_s refers to the final solution of the s-th run within the corresponding experiment. Tables 8.6–8.9 and Tables B.2–B.33 show *the best mean*, i.e.,:

$$\bar{y}_{i,j} = \min\{\bar{y}_{i,j,k,l} : 1 \leq k \leq B, 1 \leq l \leq NC\}. \tag{8.2}$$

Since the collected data is too extensive for cross comparison among all BCOc factors, in our study we mostly rely on $\bar{y}_{i,j}$. Therefore, close consideration needs to be taken when reporting average values of solution quality. In particular, the values of means $\bar{y}_{i,j,k,l}$ are often numerically close, which rises a suspicion in statistical and practical difference in the performance of the corresponding BCOc instances. According to statistical results, reported in the following sections, for $P||C_{max}$ the practical difference between two values is anywhere between 0.3 and half of unit time[(3)] w.r.t. dimension of the problem instance.

[(3)] A unit of time is taken as a measure of tasks lengths

Table 8.6: Best and worst average solutions found by corresponding BCOc algorithms for problem instance *Iogra100_12* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, ev_1							max, ev_1							max, ev_2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	813.35	7	20	1	68.47 ±20.76	14.50 ±4.35	21.00 ±0.68	**812.58**	7	20	96	68.18 ±19.02	35.90 ±9.99	52.70 ±0.84	812.39	7	20	81	70.90 ±18.88	35.60 ±9.51	50.50 ±0.89
	w	831.21	117	1	1	94.56 ±6.94	0.93 ±0.35	1.06 ± 0.37	834.02	142	2	77	94.82 ±6.88	3.38 ±0.53	3.59 ± 0.49	831.21	117	1	1	94.56 ±6.94	1.04 ±0.28	1.14 ± 0.38
p_b^1	b	813.35	7	20	1	68.47 ±20.76	14.50 ±4.39	21.00 ±0.84	812.88	5	20	5	76.55 ±16.05	17.80 ±3.67	23.20 ±0.70	812.43	7	20	8	77.96 ±14.01	19.70 ±3.49	25.30 ±0.75
	w	831.21	117	1	1	94.56 ±6.94	1.01 ±0.26	1.07 ± 0.32	1081.97	435	18	77	99.91 ±0.90	30.90 ±0.84	30.90 ± 0.86	958.53	300	19	98	99.65 ±2.57	60.80 ±1.99	61.00 ± 1.31
p_b^2	b	811.80	25	20	97	76.15 ±18.51	43.00 ±10.70	56.90 ±1.42	813.13	7	20	2	69.66 ±18.52	15.20 ±4.04	21.70 ±0.61	812.52	7	20	6	81.11 ±13.96	19.40 ±3.38	23.90 ±0.72
	w	831.21	117	1	1	94.56 ±6.94	1.01 ±0.33	1.07 ± 0.38	1117.42	435	20	77	100.00 ±0.00	32.30 ±0.82	32.30 ± 0.82	985.78	296	18	97	100.00 ±0.00	58.20 ±1.11	58.20 ± 1.11
p_b^3	b	813.35	7	20	1	68.47 ±20.76	14.50 ±4.41	21.10 ±0.71	813.35	7	20	1	68.47 ±20.76	14.60 ±4.49	21.20 ±0.66	813.33	7	20	20	70.88 ±19.68	20.40 ±5.68	28.70 ±0.78
	w	831.21	117	1	1	94.56 ±6.94	1.01 ±0.26	1.11 ± 0.31	848.18	143	2	96	98.00 ±4.10	3.97 ±0.43	4.05 ± 0.38	831.21	117	1	1	94.56 ±6.94	1.01 ±0.41	1.09 ± 0.40
p_b^4	b	813.35	7	20	1	68.47 ±20.76	14.60 ±4.41	21.30 ±0.66	812.66	5	20	12	77.96 ±13.53	20.50 ±3.68	26.30 ±0.67	**812.07**	8	20	47	82.21 ±14.00	33.70 ±5.87	41.00 ±0.83
	w	831.21	117	1	1	94.56 ±6.94	1.02 ±0.20	1.07 ± 0.29	939.29	300	7	70	99.99 ±0.10	12.80 ±0.48	12.80 ± 0.48	857.82	173	4	83	97.59 ±4.33	9.70 ±0.57	9.92 ± 0.37
p_b^5	b	813.35	7	20	1	68.47 ±20.76	14.50 ±4.42	21.10 ±0.68	813.07	5	20	6	79.00 ±16.02	18.90 ±3.82	23.90 ±0.67	812.57	8	20	8	73.26 ±16.68	18.70 ±4.20	25.40 ±0.62
	w	831.21	117	1	1	94.56 ±6.94	1.01 ±0.26	1.09 ± 0.32	1078.52	451	16	54	100.00 ±0.00	26.30 ±0.77	26.30 ± 0.77	953.88	315	17	97	99.47 ±2.83	59.30 ±1.99	59.60 ± 1.05
p_b^6	b	813.35	7	20	1	68.47 ±20.76	14.50 ±4.37	21.10 ±0.62	812.73	7	20	8	76.36 ±13.71	19.40 ±3.48	25.20 ±0.67	812.13	7	20	9	76.21 ±16.03	20.30 ±4.20	26.50 ±0.67
	w	831.21	117	1	1	94.56 ±6.94	0.97 ±0.30	1.04 ± 0.28	956.55	393	5	100	100.00 ±0.00	10.50 ±0.61	10.50 ± 0.61	879.90	258	7	98	99.29 ±2.20	22.30 ±0.80	22.40 ± 0.65
p_b^7	b	813.35	7	20	1	68.47 ±20.76	14.50 ±4.46	21.10 ±0.60	812.64	5	20	10	70.67 ±17.23	18.40 ±4.47	25.90 ±0.67	812.54	6	20	10	72.80 ±18.73	19.20 ±4.99	26.40 ±0.74
	w	831.21	117	1	1	94.56 ±6.94	1.00 ±0.24	1.06 ± 0.24	831.21	117	1	1	94.56 ±6.94	0.99 ±0.26	1.06 ± 0.31	831.21	117	1	1	94.56 ±6.94	0.93 ±0.29	0.99 ± 0.22
p_b^8	b	***810.29**	19	20	97	74.64 ±20.06	45.10 ±12.00	61.00 ±1.49	813.05	7	18	3	77.27 ±15.81	15.60 ±3.20	20.10 ±0.63	812.46	6	20	4	73.86 ±16.78	17.10 ±4.01	23.20 ±0.74
	w	831.21	117	1	1	94.56 ±6.94	1.02 ±0.24	1.06 ± 0.24	1114.98	400	20	60	100.00 ±0.00	31.90 ±0.83	31.90 ± 0.83	988.93	296	19	91	100.00 ±0.00	67.30 ±1.35	67.30 ± 1.35
p_b^9	b	813.35	7	20	1	68.47 ±20.76	14.60 ±4.42	21.20 ±0.61	813.00	6	20	4	76.52 ±17.77	18.00 ±4.08	23.40 ±0.69	812.49	7	20	8	75.95 ±17.48	20.40 ±4.71	26.90 ±0.70
	w	831.21	117	1	1	94.56 ±6.94	1.03 ±0.30	1.06 ± 0.31	1080.91	444	20	65	99.99 ±0.10	37.90 ±0.97	37.90 ± 0.97	954.09	298	18	96	99.83 ±1.07	76.60 ±1.31	76.70 ± 1.10

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table 8.7: Best and worst average solutions found by corresponding BCOc algorithms for problem instance *Iogra100_16* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, ev_1							max, ev_1							max, ev_2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	809.55	4	20	1	64.74 ±20.64	16.40 ±5.24	25.30 ±1.03	**808.77**	4	19	90	68.09 ±19.85	37.90 ±11.20	55.90 ±0.98	808.74	5	19	60	65.70 ±20.02	31.60 ±9.56	48.10 ±1.00
	w	903.25	298	1	1	99.64 ±1.69	1.19 ±0.42	1.19 ± 0.42	903.25	298	1	1	99.64 ±1.69	1.21 ±0.41	1.21 ± 0.41	903.25	298	1	1	99.64 ±1.69	1.17 ±0.42	1.18 ± 0.41
p_b^1	b	805.78	6	20	96	63.86 ±21.68	47.60 ±15.90	74.80 ±1.39	809.09	5	20	3	72.10 ±16.91	19.50 ±4.65	27.00 ±1.07	808.88	4	19	6	73.33 ±15.30	20.00 ±4.22	27.20 ±0.93
	w	903.25	298	1	1	99.64 ±1.69	1.12 ±0.41	1.12 ± 0.41	988.51	446	5	80	99.97 ±0.30	9.57 ±0.64	9.57 ± 0.64	903.25	298	1	1	99.64 ±1.69	1.13 ±0.48	1.13 ± 0.48
p_b^2	b	***805.55**	7	20	92	60.94 ±22.46	43.70 ±16.10	72.00 ±1.19	809.13	6	19	3	77.09 ±15.69	19.10 ±4.02	24.70 ±0.93	808.89	4	20	5	71.64 ±17.22	20.20 ±4.91	28.20 ±1.04
	w	903.25	298	1	1	99.64 ±1.69	1.16 ±0.44	1.17 ± 0.45	1001.00	496	10	61	99.94 ±0.60	17.40 ±0.65	17.40 ± 0.65	911.74	417	6	98	98.45 ±5.01	21.10 ±1.18	21.40 ± 0.63
p_b^3	b	809.21	5	19	2	66.27 ±17.77	16.10 ±4.40	24.20 ±0.96	809.21	5	19	4	69.67 ±16.94	17.70 ±4.39	25.40 ±0.86	809.00	5	20	35	68.38 ±20.75	28.10 ±8.47	41.10 ±1.07
	w	903.25	298	1	1	99.64 ±1.69	1.11 ±0.37	1.12 ± 0.38	909.13	324	2	92	98.99 ±4.14	4.39 ±0.68	4.46 ± 0.62	903.25	298	1	1	99.64 ±1.69	1.12 ±0.41	1.12 ± 0.41
p_b^4	b	809.55	4	20	1	64.74 ±20.64	16.70 ±5.22	25.70 ±1.05	809.06	4	20	11	77.26 ±13.43	23.90 ±4.24	30.90 ±1.20	**808.68**	5	20	94	75.74 ±15.31	49.80 ±10.00	65.80 ±1.25
	w	903.25	298	1	1	99.64 ±1.69	1.23 ±0.51	1.23 ± 0.51	920.59	298	2	60	99.24 ±2.82	3.73 ±0.47	3.78 ± 0.44	903.25	298	1	1	99.64 ±1.69	1.13 ±0.42	1.14 ± 0.42
p_b^5	b	805.57	7	20	97	60.06 ±23.21	48.50 ±18.80	80.90 ±1.41	809.15	4	20	5	75.12 ±16.47	21.20 ±4.61	28.30 ±1.17	808.98	5	20	7	74.20 ±16.98	22.40 ±5.10	30.10 ±0.91
	w	903.25	298	1	1	99.64 ±1.69	1.15 ±0.38	1.15 ± 0.38	985.59	443	5	80	99.66 ±1.72	9.80 ±0.53	9.83 ± 0.53	904.98	282	2	86	98.43 ±5.09	6.58 ±0.59	6.68 ± 0.49
p_b^6	b	808.68	25	20	96	70.99 ±20.23	55.00 ±15.70	77.50 ±1.17	809.01	6	20	6	74.64 ±14.61	21.90 ±4.24	29.20 ±1.11	808.81	4	20	8	73.95 ±17.79	23.20 ±5.57	31.30 ±0.96
	w	903.25	298	1	1	99.64 ±1.69	1.25 ±0.46	1.25 ± 0.46	942.51	349	2	60	99.73 ±1.29	3.89 ±0.34	3.90 ± 0.33	909.65	278	2	99	98.62 ±4.18	6.80 ±0.55	6.90 ± 0.44
p_b^7	b	809.18	4	20	56	64.04 ±20.22	35.30 ±11.10	55.10 ±1.01	808.83	5	19	85	63.18 ±21.98	40.50 ±14.20	64.20 ±1.10	808.89	5	20	69	63.81 ±20.04	40.10 ±12.50	62.80 ±1.21
	w	903.25	298	1	1	99.64 ±1.69	1.18 ±0.41	1.18 ± 0.41	903.25	298	1	1	99.64 ±1.69	1.16 ±0.39	1.16 ± 0.39	903.25	298	1	1	99.64 ±1.69	1.15 ±0.43	1.15 ± 0.43
p_b^8	b	805.82	7	19	94	61.56 ±21.94	44.10 ±15.70	72.30 ±1.44	809.16	5	19	3	74.69 ±16.72	18.70 ±4.14	25.10 ±1.04	808.88	5	20	3	73.11 ±17.85	20.20 ±4.87	27.50 ±0.92
	w	903.25	298	1	1	99.64 ±1.69	1.22 ±0.44	1.23 ± 0.44	1005.06	404	8	72	99.91 ±0.90	14.90 ±0.64	14.90 ± 0.64	907.09	308	11	98	98.64 ±4.45	43.30 ±1.88	43.90 ± 0.99
p_b^9	b	805.69	6	20	97	63.35 ±22.95	59.40 ±21.70	94.00 ±1.61	809.18	4	19	5	74.23 ±14.39	20.20 ±3.95	27.10 ±1.09	808.91	4	20	4	69.49 ±16.64	19.80 ±4.65	28.50 ±0.93
	w	903.25	298	1	1	99.64 ±1.69	1.21 ±0.41	1.21 ± 0.41	984.40	408	5	77	99.83 ±1.23	10.30 ±0.60	10.30 ± 0.60	903.25	298	1	1	99.64 ±1.69	1.12 ±0.43	1.13 ± 0.44

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table 8.8: Best and worst average solutions found by corresponding BCOc algorithms for problem instance *Iogra100_12* [Dav06b]. Stopping criterion, $T = 0.1[s]$.

		min, ev_1							max, ev_1							max, ev_2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	810.99	5	1	1	5008.44 ±2482.99	52.50 ±26.00	9589.42 ±177.95	809.87	6	5	7	1051.46 ±408.15	61.90 ±23.90	1706.47 ±20.36	809.93	5	5	6	960.04 ±438.50	56.10 ±25.60	1718.40 ±19.13
	w	815.69	6	19	91	127.82 ±53.15	62.10 ±25.80	206.89 ± 2.80	811.73	5	19	81	129.90 ±47.87	60.40 ±22.40	216.88 ± 2.67	811.86	4	18	100	125.07 ±43.55	63.10 ±22.00	199.16 ± 2.53
p_b^1	b	810.03	12	10	98	235.70 ±93.17	62.60 ±24.90	377.53 ±5.15	809.01	5	5	25	1029.44 ±303.42	69.10 ±20.20	1492.62 ±18.03	809.13	6	7	12	660.77 ±238.21	62.00 ±22.40	1071.04 ±14.87
	w	826.18	17	18	22	143.32 ±117.10	40.20 ±32.70	358.35 ± 5.12	905.93	287	20	95	270.38 ±11.82	99.40 ±3.50	274.23 ± 5.34	879.28	198	20	97	151.53 ±11.95	99.30 ±7.56	154.10 ± 2.35
p_b^2	b	808.17	11	15	97	160.51 ±52.44	66.10 ±22.10	241.68 ±4.00	**808.47**	3	7	25	804.91 ±207.95	72.50 ±18.80	1114.09 ±15.49	**808.85**	5	11	9	460.73 ±162.89	63.70 ±22.60	725.46 ±8.92
	w	826.81	23	20	15	98.56 ±113.08	27.80 ±31.90	357.54 ± 5.68	918.74	242	20	93	295.31 ±9.39	100.00 ±2.24	296.71 ± 5.07	904.07	211	20	99	150.43 ±5.57	101.00 ±3.50	151.02 ± 2.17
p_b^3	b	810.68	6	14	2	400.89 ±157.91	59.50 ±23.50	675.54 ±8.69	810.69	5	13	2	437.92 ±168.24	61.40 ±23.70	716.28 ±8.85	810.71	7	16	2	367.76 ±125.70	63.80 ±22.00	579.26 ±6.61
	w	813.65	5	19	96	140.03 ±36.58	70.80 ±18.60	199.42 ± 3.03	812.78	5	20	95	133.08 ±34.53	70.00 ±18.60	190.53 ± 2.43	812.78	6	19	90	124.42 ±38.61	63.60 ±19.90	196.96 ± 2.20
p_b^4	b	810.98	5	11	1	469.11 ±231.01	53.90 ±26.40	875.36 ±10.11	809.54	6	5	25	905.02 ±313.71	64.00 ±22.10	1419.52 ±17.27	809.38	4	6	45	564.31 ±187.99	65.10 ±21.80	872.10 ±11.76
	w	825.67	16	19	89	96.97 ±62.55	48.90 ±31.50	200.33 ± 2.92	819.29	89	20	100	189.68 ±19.38	92.50 ±9.16	206.46 ± 3.15	811.46	6	1	95	3382.17 ±1558.79	55.40 ±25.50	6148.05 ± 98.55
p_b^5	b	810.60	10	9	100	232.57 ±95.11	59.50 ±24.40	392.25 ±5.30	809.01	4	6	25	829.64 ±234.13	68.60 ±19.40	1213.55 ±13.91	809.19	6	13	20	344.69 ±101.17	71.00 ±21.00	487.90 ±5.69
	w	827.05	22	19	40	116.12 ±86.72	44.80 ±33.40	262.94 ± 4.10	911.23	312	20	99	250.28 ±16.71	98.80 ±5.92	255.56 ± 4.44	894.43	256	20	100	136.78 ±4.56	100.00 ±2.66	138.10 ± 2.11
p_b^6	b	810.99	5	5	1	1020.19 ±499.55	53.00 ±26.00	1930.43 ±22.64	809.44	5	7	9	771.97 ±255.43	68.40 ±22.60	1133.40 ±12.76	809.18	6	10	24	405.38 ±117.85	68.10 ±19.80	598.41 ±7.34
	w	826.55	25	20	81	71.22 ±54.87	41.20 ±31.80	174.85 ± 2.69	837.64	156	20	99	202.07 ±15.82	97.20 ±7.25	209.56 ± 3.31	814.41	105	20	100	133.10 ±13.82	90.90 ±9.20	147.52 ± 2.84
p_b^7	b	810.99	5	1	1	4991.86 ±2488.48	52.30 ±26.10	9580.42 ±156.67	809.93	5	4	8	1256.01 ±492.88	61.90 ±24.40	2037.92 ±24.77	809.95	5	6	9	818.78 ±331.21	62.60 ±25.40	1311.05 ±14.29
	w	813.82	6	20	84	98.07 ±38.92	58.20 ±23.20	169.54 ± 2.30	812.63	6	20	99	99.17 ±33.29	65.50 ±22.10	152.65 ± 2.12	812.74	5	17	100	108.72 ±38.70	62.50 ±22.30	175.23 ± 2.23
p_b^8	b	***807.61**	12	16	97	139.83 ±45.10	64.40 ±21.30	216.10 ±4.04	808.63	5	7	25	738.67 ±182.88	68.10 ±16.90	1089.86 ±14.62	**808.85**	4	12	9	415.30 ±138.42	64.60 ±21.40	645.62 ±7.85
	w	824.26	23	20	15	154.06 ±109.88	45.00 ±32.20	343.49 ± 5.45	934.90	312	20	99	269.70 ±13.12	99.90 ±4.30	272.62 ± 4.27	926.40	275	20	94	136.43 ±2.64	101.00 ±0.87	136.67 ± 2.27
p_b^9	b	811.00	5	1	1	4912.53 ±2465.86	51.40 ±25.80	9586.96 ±173.79	809.20	4	5	32	872.01 ±256.69	66.30 ±19.60	1318.98 ±17.43	809.38	6	6	10	722.33 ±288.90	60.20 ±24.00	1204.38 ±14.12
	w	826.87	30	20	43	96.16 ±69.36	47.00 ±33.80	207.41 ± 3.31	934.36	265	20	87	245.71 ±8.28	100.00 ±2.88	247.57 ± 3.58	928.34	240	20	97	115.30 ±2.73	101.00 ±1.85	115.72 ± 1.68

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table 8.9: Best and worst average solutions found by corresponding BCOc algorithms for problem instance *Iogra100_16* [Dav06b]. Stopping criterion, $T = 0.1[s]$.

		min, ev_1							max, ev_1							max, ev_2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	807.73	4	2	1	2161.31 ±1093.29	50.40 ±25.50	4305.81 ±53.99	807.14	4	4	9	1101.85 ±496.42	59.10 ±26.60	1871.01 ±23.56	807.20	4	9	3	477.65 ±216.13	53.40 ±24.10	899.12 ±10.71
	w	809.78	5	20	79	98.45 ±43.54	55.80 ±24.80	178.14 ± 3.70	808.49	4	19	100	100.95 ±38.74	59.30 ±22.90	171.47 ± 4.00	808.58	3	19	100	89.45 ±36.05	55.80 ±22.50	161.66 ± 2.63
p_b^1	b	805.15	6	20	99	78.45 ±32.26	59.00 ±24.20	133.69 ±2.59	806.92	2	5	20	861.16 ±278.22	63.70 ±20.70	1358.10 ±19.35	806.96	3	5	8	774.78 ±320.70	53.70 ±22.30	1448.15 ±14.57
	w	812.18	8	20	13	160.16 ±76.44	52.60 ±24.80	307.66 ± 10.59	846.91	305	17	81	242.38 ±38.17	88.80 ±13.30	274.06 ± 7.94	829.19	267	20	99	117.62 ±15.45	90.60 ±11.60	130.96 ± 2.29
p_b^2	b	***804.97**	5	14	96	113.43 ±47.27	54.50 ±22.80	208.55 ±3.27	**806.83**	3	6	6	861.73 ±292.45	66.20 ±22.40	1306.18 ±15.27	**806.64**	3	2	82	992.72 ±371.61	60.60 ±22.90	1645.54 ±17.73
	w	811.86	7	20	6	163.03 ±85.72	46.10 ±24.00	355.27 ± 14.34	876.32	358	20	81	230.11 ±22.72	95.00 ±8.21	243.96 ± 6.87	851.57	215	20	99	124.94 ±12.49	95.30 ±9.16	132.28 ± 2.11
p_b^3	b	807.49	3	4	3	1107.24 ±533.88	53.30 ±25.80	2086.37 ±23.80	807.46	3	6	9	670.11 ±307.68	53.70 ±24.70	1252.33 ±14.89	807.46	3	9	3	472.30 ±209.33	51.90 ±23.00	913.63 ±12.85
	w	809.51	4	20	93	104.12 ±34.26	63.30 ±20.50	165.99 ± 3.50	808.97	3	16	86	136.16 ±45.21	61.60 ±20.90	220.83 ± 3.24	808.87	5	20	95	102.74 ±29.48	66.50 ±19.00	155.31 ± 2.93
p_b^4	b	807.72	4	3	1	1475.97 ±737.21	51.10 ±25.30	2897.49 ±38.81	807.06	4	4	23	969.51 ±369.87	60.10 ±23.10	1616.79 ±17.17	807.02	3	7	9	560.22 ±248.19	54.70 ±24.30	1026.91 ±13.73
	w	812.06	11	20	75	103.61 ±41.74	60.30 ±24.40	173.27 ± 3.76	811.69	148	18	97	163.68 ±28.76	79.70 ±14.20	206.24 ± 4.26	808.31	4	20	72	118.72 ±35.55	70.20 ±20.90	170.24 ± 3.07
p_b^5	b	805.26	6	20	97	69.02 ±28.45	54.90 ±22.60	126.31 ±2.63	806.89	4	4	20	1030.72 ±409.66	61.50 ±24.40	1682.03 ±22.16	807.00	4	8	8	533.48 ±216.91	60.30 ±24.60	889.41 ±10.50
	w	812.51	8	20	15	146.65 ±70.26	50.50 ±24.20	292.05 ± 9.17	856.75	364	20	88	198.88 ±23.67	92.70 ±10.30	215.75 ± 5.80	832.59	173	18	99	121.93 ±17.07	90.10 ±12.30	136.32 ± 2.10
p_b^6	b	807.72	4	3	1	1470.95 ±742.09	51.00 ±25.80	2894.18 ±37.14	807.03	4	6	9	689.96 ±265.81	57.90 ±22.40	1197.49 ±14.52	807.03	4	6	4	761.41 ±337.79	59.10 ±26.10	1295.69 ±16.19
	w	811.86	7	19	40	118.54 ±53.78	54.50 ±24.80	219.70 ± 5.65	814.12	129	18	98	168.62 ±30.78	83.30 ±15.10	203.37 ± 4.23	811.54	137	20	93	105.25 ±23.97	79.30 ±18.10	133.74 ± 2.48
p_b^7	b	807.73	4	3	1	1461.36 ±729.35	50.70 ±25.20	2896.70 ±42.36	807.17	4	6	5	667.31 ±311.51	51.70 ±24.20	1292.36 ±16.48	807.22	4	14	3	304.10 ±137.58	54.40 ±24.70	561.44 ±7.22
	w	809.20	4	17	90	98.34 ±41.62	57.50 ±24.20	172.46 ± 2.73	808.96	5	12	98	127.77 ±56.86	54.10 ±24.00	237.42 ± 2.67	808.93	4	15	99	101.47 ±39.00	57.30 ±22.00	178.19 ± 2.02
p_b^8	b	805.12	6	18	93	77.39 ±35.66	51.90 ±24.00	150.11 ±2.87	806.85	3	5	6	911.13 ±333.84	58.80 ±21.50	1556.32 ±18.72	806.82	4	8	8	547.26 ±212.06	61.80 ±24.10	887.77 ±10.80
	w	811.52	8	20	7	173.71 ±91.30	50.80 ±26.50	342.70 ± 12.04	874.77	317	18	100	233.80 ±26.89	93.60 ±9.84	251.32 ± 7.14	871.47	261	19	100	118.03 ±11.32	96.10 ±8.82	123.99 ± 1.99
p_b^9	b	805.59	6	20	97	65.34 ±24.49	61.70 ±23.30	106.87 ±1.99	807.08	2	5	20	796.84 ±270.75	63.20 ±21.60	1266.73 ±16.18	807.07	4	5	8	814.03 ±331.63	59.90 ±24.50	1366.67 ±16.06
	w	812.62	7	20	12	152.94 ±70.93	52.40 ±24.10	292.67 ± 8.96	858.43	374	20	100	177.92 ±20.95	93.20 ±10.40	191.96 ± 4.97	844.04	215	20	100	94.85 ±9.06	94.80 ±8.84	101.03 ± 1.55

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

8.4 Screening BCOc parameters: basic plots

Results in Tables 8.6–8.9 and Tables B.2–B.33 (referred to as the *basic tables* in the remainder of this thesis) show that none of the BCOc instances is able to reach the optimal solution, regardless of the stopping criterion. Instead, a set of solution's quality values (hereafter referred to as a *sample* or $\{y_s : s = \overline{1, n_{run}}\}$) produced by a single experiment exhibits a bell-shaped distribution resembling the most to the normal (Gaussian) distribution. We address this subject rigorously in Section A.3.2 (see Fig. A.7). We find that statistics of central tendency for solution quality samples provide a satisfactory measure to address questions of sensitivity analysis of BCOc. In addition, the basic tables have already estimated the best parameter configurations of BCOc for each considered problem instance.

According to [Rid07, pg. 23] at this stage of the empirical study it is common to perform *screening* of BCOc parameters. Screening represents a method to determine which parameters affect BCOc performance. The approach we present here is based on 3-D and 2-D *basic plots* established on the solution quality means, i.e, values of $\bar{y}_{i,j,k,l}$ for each $i \in \mathcal{L}$ and $j \in \mathcal{M}$.

8.4.1 3-D plots: qualitative parameters

We assess and compare response surface plots of parameters B and NC for different values of qualitative factors Lp and ME[(4)] (Figs. 8.2, B.5 and B.6). The response surface plots are able to demonstrate any tendencies towards local optimum of the parameter configuration sub-space and help to make a more informed decision about the future course of the empirical study. Namely, the valleys in the landscape mark the area of parameter space that yield the best results. However, effect of parameter ME is hard to identify from these figures. One way to address this issue is demonstrated in Fig. 8.3. All graphics in the figure are generated w.r.t. problem instance and the levels of parameters Lp and ME. The graphics plot percentage relative errors (in remainder referred to as simply relative errors) of the best means, i.e., of $\bar{y}_{i,j}$. The profile lines, especially for problem instances $n = \{100, 150, 200, 250\}$, suggest interactions between BCOc parameters ME and Lp (see [How10, pg. 421] for definition of interactions). What we consider under interaction is the intersection of profile lines for one problem instance along levels of Lp. It is worth noting that utilized design points in our experimental study do not follow a common methodology, e.g., marginal means or a random design technique. Consequently, by using $\bar{y}_{i,j}$ points we believe that observable effect of ME is smaller against employing other experimental designs from the literature. The method $\min(ev_1)$ exhibits satisfying performance for two instances of class $m = 16$, depending on the level of parameter Lp. Additionally, $\max(ev_2)$ shows to be better than $\max(ev_1)$ for problem instance Iogra100_12. Nevertheless, we are not certain about the statistical difference between the corresponding results $\bar{y}_{i,j}$, especially for the rest of problem instances.

Beside the established best configurations of B and NC in the basic tables, from Figs. 8.2, B.5 and B.6 we may conclude several results. Namely, from 3-D basic plots (Figs. 8.2, B.5 and B.6) the improvement in the solution quality is established for larger

[(4)] We refer to each value of parameter ME or Lp, as its *level*. A set of data samples of one parameter, generated at a level of the other, is referred to as a *group* (see pg. 261).

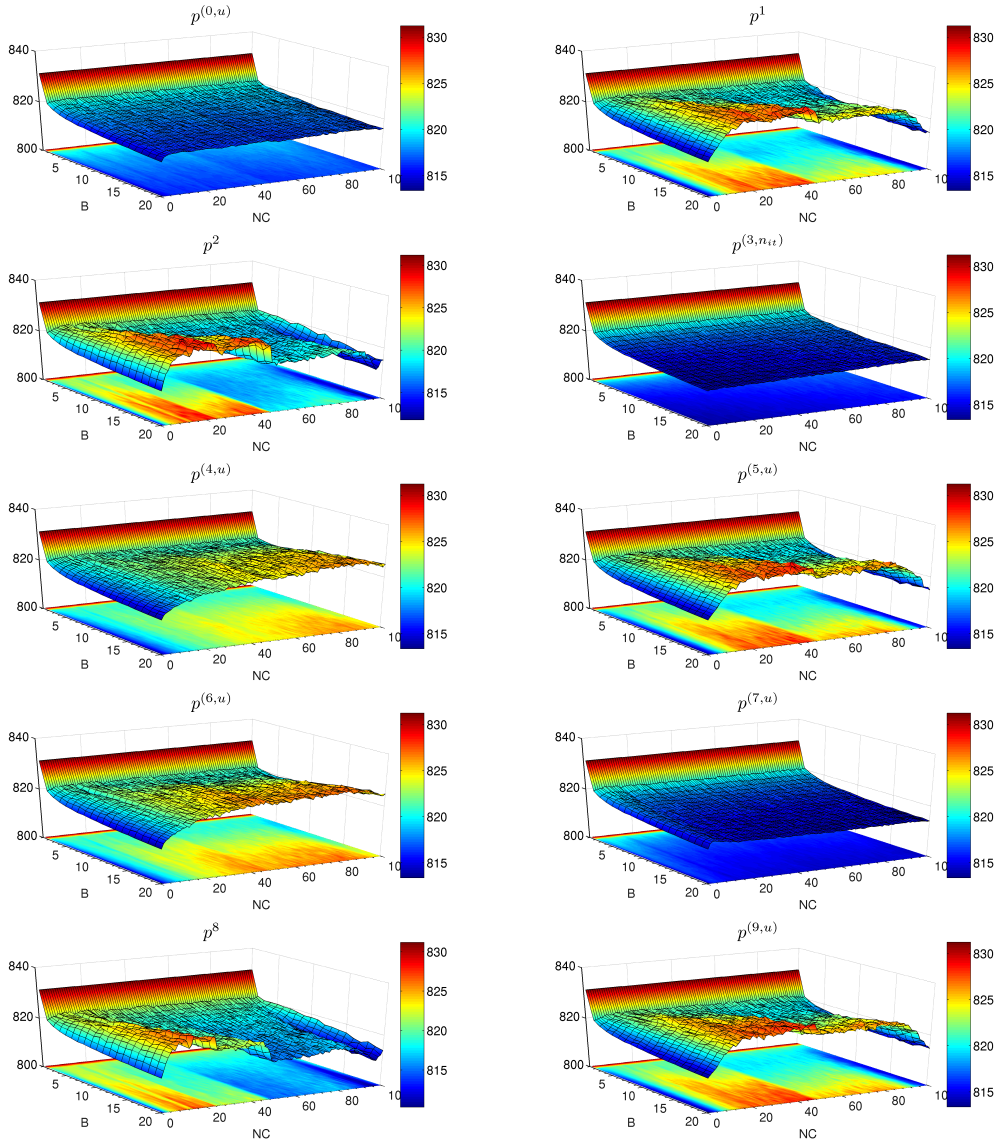


Figure 8.2: Response landscape of 10 BCOc with $\min(ev_1)$ for Iogra100_12, $N_{it} = 100$.

B, however, either for smaller values of NC or the complete domain $\mathcal{NC}$ and the particular loyalty function. For example, a linear improvement of solution quality is recognized in case of $p_b^0, \min(ev_1)$ on the restricted domain of $\mathcal{NC}$, while for the $p^{1,2,8}$ the solution quality degrades significantly when $NC \leq 50$, as the population of bees is growing. Therefore, the increase in the population of bees doesn't necessarily lead to increase in the solution quality. Such behavior indicates high nonlinearity and complex interaction between algorithm's parameters, observed in Fig. 8.3. In addition, high nonlinearity observed in surface plots on the $\mathcal{B} \times \mathcal{NC}$ space is detected for loyalty functions of Class II. Regarding the general conclusions, method $\max(ev_1)$ (i.e., ME_2) has

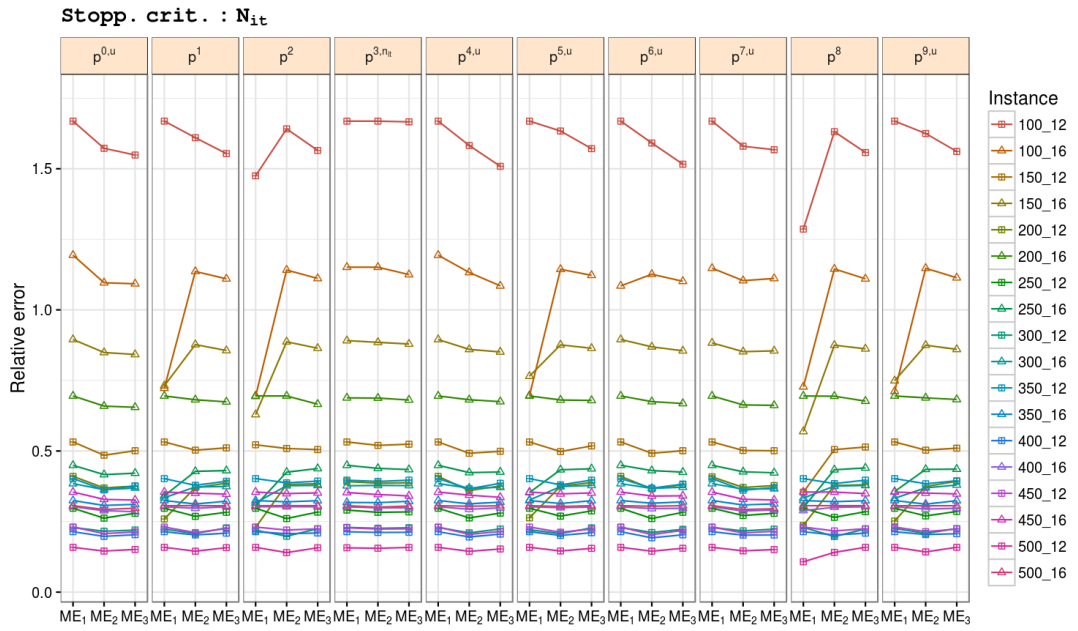


Figure 8.3: Effects of parameters ME, Lp and the problem instance characteristics. Profile lines with squares denote problem instances in the class $m = 12$ and with triangles the class $m = 16$. Different colors refer to value of n. Stopping criterion is $N_{it} = 100$.

produced the largest span among $\bar{y}_{i,j,k,l}$ values, indicated by the shape of the response landscape in Fig. B.5. Method $\max(ev_2)$ (i.e. ME_3) induces less variability, however, lacking the practical improvement (Fig. B.6). Method $\min(ev_1)$ (i.e., ME_1) generates the smallest variability (Fig. 8.2) and produces improvements in the solution quality for specific values of B and NC.

Unlike Figs. 8.2, B.5 and B.6, the relations between qualitative factors are more distinct if we compare surface plots on one graphic. We address this in Section B.1.3, (pg. 267). In Fig. B.7 results for $p^{0,u}, p^{1,2,8}$, at the level of ME corresponding to $\min(ev_1)$, are compared. Plots are generated for relative errors. The comparison of the loyalty functions indicate that average performance of BCOc, while utilizing p_b^8, for $B \geq 15$ and $95 \leq NC \leq 98$, is better against other three loyalty functions. We are motivated to visualize experimental results, however, comparing all the values of the parameter Lp on a single figure is obviously unmanageable. Therefore, graphical representation are in the remainder of the chaper founded on 2-D plots.

At this point we may address a couple of research question, e.g., Q_1 and Q_3. Regarding the former, parameters Lp and ME exibit the highest influence and possible interconnection. The finding is based on two observations: (a) difference between $\bar{y}$ of the best and $\bar{y}$ of the worst experiment is the largest under their influence; (b) success of several loyalty functions is related to values of ME. Regarding the question of robustness (Q_3) Fig. 8.3 illustrates that the problem structure exhibits high impact on the BCOc performance. The success of the particular combination of qualitative parameters cannot be easily identified in Fig. 8.3 since none was able to consistently outperform the other on the complete problem set. The only direction is to observe effect of one

parameter at the level of the other.

8.4.2 2-D plots: quantitative parameters

Here, we further exploit graphical analysis of the BCOc parameters. The results are presented in Fig. 8.4. The figure illustrates influence of BCOc parameters on the reported average solutions quality (relative error), for two problem instances. In particular, to maintain an overview on the influence of problem instances, we eleminated one parameter to generate 2-D graphics. According to basic tables we omit parameter B because the most experimental runs reported similar values for its best configurations. Therefore, each graphic consists of set of plots that reveal the influence of loyalty function with regard to method of evaluation, for a fixed value B and different values of NC.

The main objective of this presentation is to visually inspect improvements in the solution quality when parameter NC changes its value w.r.t. the structural parameters of BCOc. Graphics are arranged to distinguish influence of ME and loyalty functions when $NC \in [1, 100]$. Each profile line in Fig. 8.4 corresponds to the value of B that has produced the best results. Namely, B takes values from $[18, 20]$ among the profile lines. For example, for Iogra100_12 and $\min(ev_1)$, the profiles of the best results correspond to $B = 20$. On the same instance, in the case of configuration $\max(ev_1)$, $p^{9,u}$, the best average result is achieved when $B = 18$. The color of the plot corresponds to a single loyalty function, whereas dashed black line signifies BCOc reference case. Reference case is used to simulate the behaviour of an underlying heuristic, i.e., instead of executing `sLPT+BF` outside the BCOc frame, the results are obtained by running BCOc algorithm for $NC = 1, B = 20$. The difference between the corresponding configuration of BCOc and the standalone heuristic might be exhibited for $B > 1$ due to overhead caused by evaluation phase of backward pass. However, a time difference is practically insignificant for the case of $N_{it} = 100$. Detailed elaboration about this approximation of the reference case is given in Section (A.2.2) (pg. 258).

We draw a few compelling conclusions from Fig. 8.4. Firstly, for some cases of loyalty functions an impact of ME is not easy to categorize as it is the case for $p^{0,3,7}$. Namely, the three functions exhibit similar properties regardless of values of ME. Beside their exploratory nature, the functions converge fast toward parts of the search space in which they dwell until end of an iteration. Therefore, it is most likely that the search gets fast stranded in one of the local minima. The lack of perturbations in solution quality as NC changes thus becomes obvious.

|

Unlike previous group of loyalty functions, others demonstrate a large variability of the solution quality. The variation depends on the utilization of evaluation function (or method of evaluation) and structure of the problem instance. Although show significant fluctuations, these loyalty functions are able to bring improvements in the solution's quality for at least one value of ME. Among them, loyalty functions p_b^2 and p_b^8 perform the best in respect to the reference case. In addition, graphics also reveal that inside of these set of loyalty functions certain groups exhibit similar behaviour. Such groups are: $p_b^{1,5,9}$, $p_b^{2,8}$ and $p_b^{4,6}$. To distinguish the influence of loyalty functions within a group, further analysis needs to be conducted on the properties of recruitment process. We investigate this in Section 8.8.

Because the results are sensitive to the choice of problem instance, it is obvious

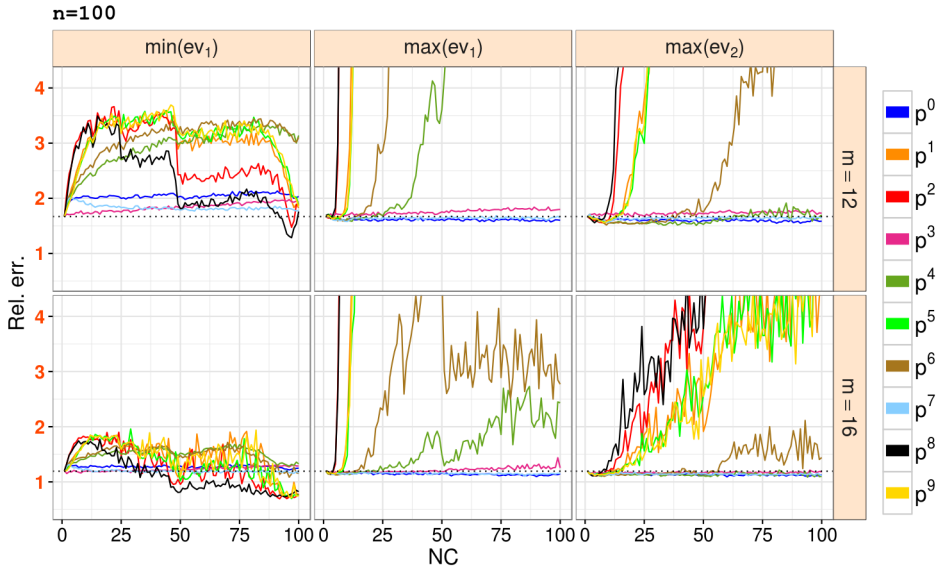


Figure 8.4: Representative graphic of influence of three BCO parameters (*ME*, *LF*, *NC*) on solution quality for $N_{it} = 100$. Graphic represents changing of average solutions over values of NC for each loyalty function, on problem instance Iogra100_12. For easiness of comparing, relative error is used.

that the corresponding visual analysis on the entire problem set should be undertaken (Fig 8.5). As previously, the value for B varies in interval $[18, 20]$ when generating good quality solutions, with one exception where $B = 15$ is reported by function p^8 on problem instance Iogra400_12. The series of graphics in Fig. 8.5 consists of Fig. 8.4 and eight more, in regard to the dimension of a problem instance. Once more it should be noted that NC values do not cover complete parameter space for problems of dimension $n > 100$ because of high computational cost. However, we can still notice similarities between the graphics from different groups, and draw similar conclusions as for $n = 100$. The first paramount result is related to Class II type of loyalty functions, such as p^k, $k \in \{0, 3, 4, 5, 6, 7, 9\}$. The loyalty functions $p^{0,u}, p^{3,it}$ and $p^{7,u}$ are the most conservative due to small changes in the reported average solutions over the complete interval $NC \in [1, 100]$, regardless of method of evaluation. Remaining Class II loyalty functions show high sensitivity to utilization of method of evaluation and problem instance. No pattern is able to be identified in respect to NC that generates high quality solutions. Actually, only p^5 and p^9 succeed to be better than the reference case for Iogra100_12/16, Iogra150_16, Iogra200_12, Iogra250_16 and Iogra300_12/16. Furthermore, these two loyalty functions exhibit similar behavior throughout the search. Loyalty functions p^1, p^2, p^8, show large sensitivity to changes of quantitative parameters B and NC and the choice of the problem instance. Between these three, the most unsuccessful is p^1 that has generated solutions resembling to solutions of $p^{5,9}$. Loyalty functions p^2 and p^8 are the only one that demonstrate a certain pattern on the inspected domain of NC which brings improvements w.r.t. the reference case.

Until this point we have dealt with the $N_{it} = 100$ case study. The new set of graphics additionally covers results determined for CPU case study, i.e., when the maximal allowed time is restricted with $T = n/100[s]$ (Fig. 8.6). The reference case w.r.t. BCOc

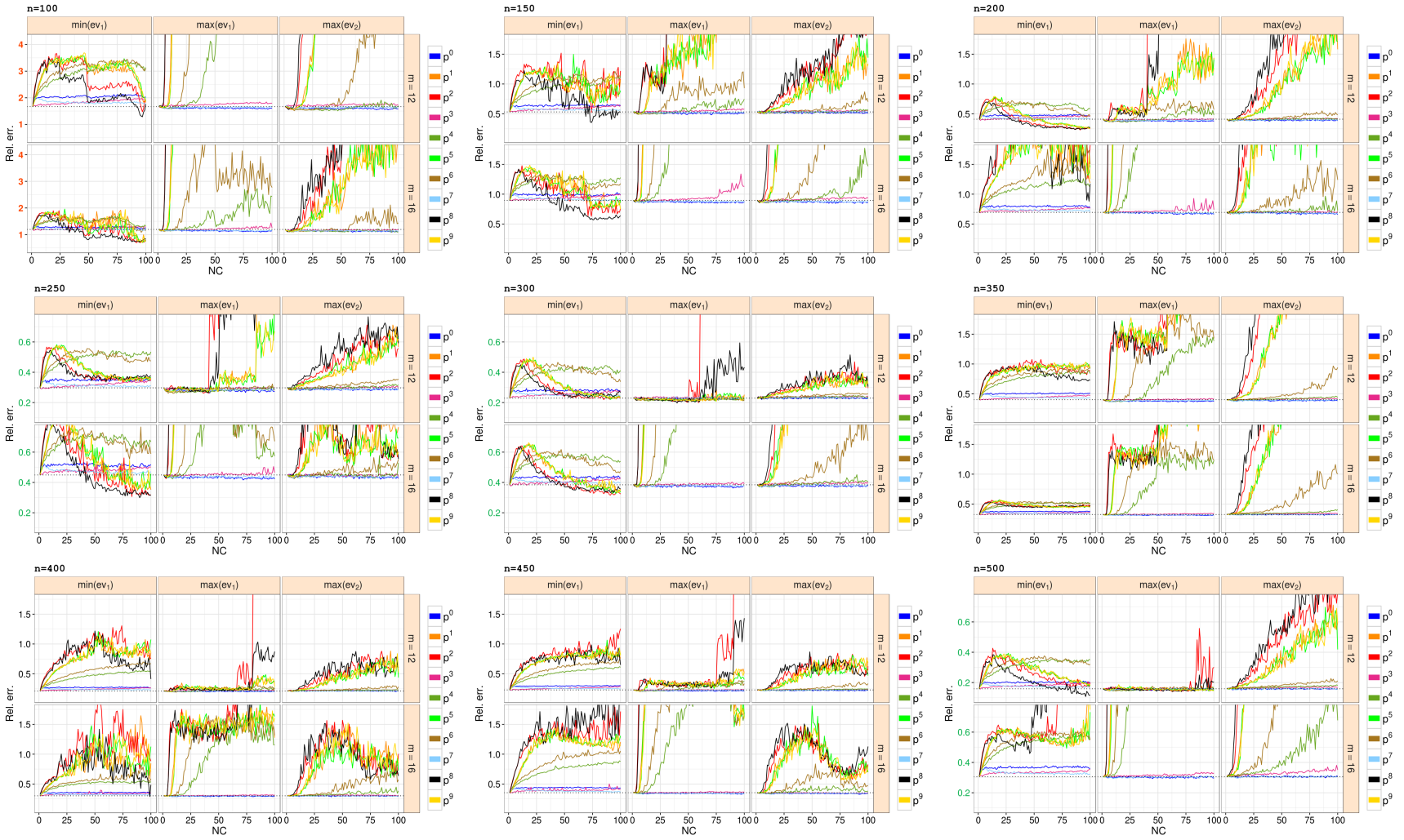


Figure 8.5: Influence of three methods of evaluation and 10 loyalty function on the performance of BCOc for $P||C_{max}$, when $NC \in [1, 100]$ and fixed B that generated the best results. Stopping criterion is maximum number of iterations, $N_{it} = 100$.

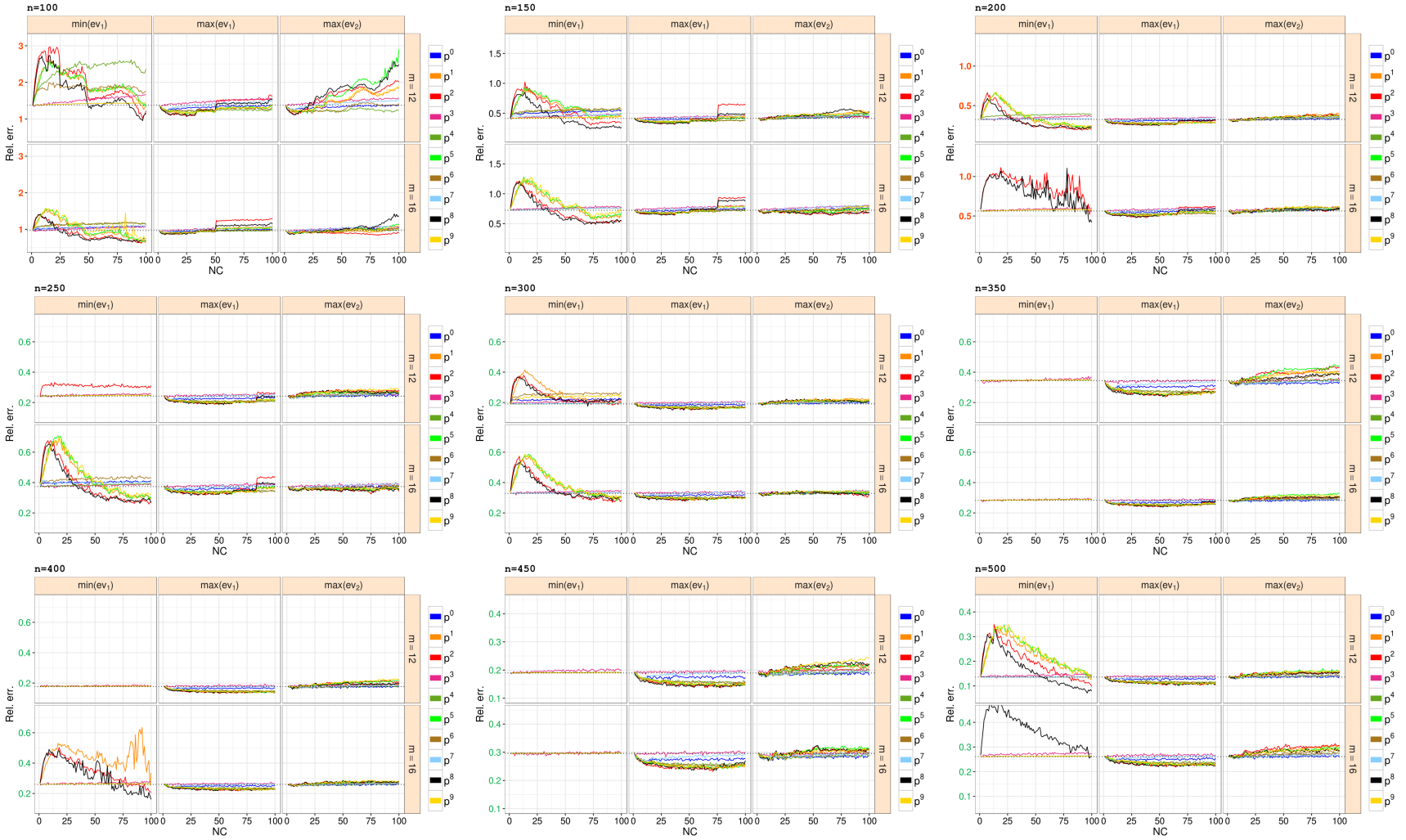


Figure 8.6: Influence of three methods of evaluation and 10 loyalty function on the performance of BCOc for $P||C_{max}$, $NC \in [1, 100]$ and fixed B that has reported the best solution. Stopping criterion is allowed CPU time $T = n/100[s]$.

parameters is here different. Namely, the performance of the sLPT+BF heuristic imply values $NC = 1$ or $B = 1$. According to the results of the experiments the solution quality of the heuristic does not change for different NC at $B = 1$. Reference case is therefore determined as the best result obtained by BCOc when $NC = 1$, $B \in [1, 20]$. Consequently, parameters Lp and ME have no impact (the backward pass does not occur for the reference case).

To explain properties of plots in Fig. 8.6, we examine results on a case to case basis. Namely, the graphics report lower variability of the response values for structural parameters. For problem instances $m = \{12, 16\} : n = \{350, 450\}$ and $m = 12 : n = 400$ the method $\min(ev_1)$ demonstrates the smallest variability. The reason is that the best results for the particular problem instances is, except for $p^{3,u}$, reported by the reference case (see basic tables). This has not happen for other parameter configurations, however, we detect that BCOc algorithms reached the ceiling effect and reported solutions of a similar quality. Disadvantage of the particular set of results in Fig. 8.6 is the range of parameter B for which graphics are generated. For example, in case of $m = 12 : n = 150$ and $\min(ev_1)$ the corresponding profile lines are produced for large range of values of B. Contrary, instances $n = \{300, 400, 500\}$ seem to be easy because sLPT+BF produced high quality results. However, we remind that NC domain is restricted and BCOc configurations for larger NC might produce better results.

The rest of our study is focused on investigation of simple effects with help of statistical tests. To conclude on the overall success of each method of evaluation (on the complete problem set), we present results of Friedman's rank and Kruskal-Wallis test in the following section. The tests contribute to our goal to investigate general behavior of the BCOc algorithm.

8.5 Statistical analysis: hypothesis testing

In previous sections, we emphasized the importance of establishing statistical and practical differences between the results on which we base our conclusions. There are several other reasons that direct the study towards methodological statistical analysis. We summarize them as follows.

- Increasing the number of figures that show solution quality response landscapes in general does not help to address our research goals. In addition, along already demanding basic tables, graphics require great deal of space.
- An Mk1[(5)] test does not reveal to which degree we consider two BCOcs different. To establish a clear difference between the best among three BCOc algorithms, the result of comparing values 1603.51, 1603.15 and 1603.52 from Table B.8 (for problem instance Iogra300_12, pg. 280), commonly depends on the practitioner making a observation.
- A number of pairwise comparisons between all BCOc instances, for a particular problem instance, is too large to be conducted without increasing *familywise* error rate [How10, pg. 151]. The error rate defines the probability of making Type I error (making false discoveries by rejecting null hypothesis when it should not be rejected) at least once during the pairwise comparison.

[(5)] Mk1 is abbreviation of *Mark One Eyeball* used in a military as a slang for visual inspection [Bus].

- Another issue concerns interactions between BCOc parameters. According to [How10], difference between any two means might originate from: different levels of a parameter or their interactions. From 8.2, B.5 and B.6 we can already detect significant interactions between parameters B and NC.

Therefore, we perform multiple comparison statistical tests that help to establish a clear difference in the performance of the BCOc instances. Due to the existence of interactions we examine simple effects ([How10, 416]) and separate the statistical analysis of BCOc performance among two case studies. We define the null hypothesis as the statement that all means are equal (in case of parametric test) or samples are from the same population (in case of non-parametric tests), however, we do not explore the complete configuration sub-space. Instead, we use carefully determined samples of data (addressed for formula 8.2, pg. 154). The first case study explores an effect of parameter Lp for outcomes achieved at one level of the parameter ME. It is based on multiple comparison of data samples (Section A.3.1.2, pg. 259) between the values of the parameter Lp. We employ fixed-model ANOVA with repeated measures (RMANOVA) for within-subject evaluation. The justification for using this parametric test is carefully elaborated in section A.3.2 (pg. 261). Moreover, to accommodate RMANOVA we utilize post-hoc tests to conclude which loyalty function has exhibited the greatest impact on the BCOc performance. The post-hoc test is conducted as a two-sided Dunnett's test with Hommel's correction. The main advantage of parametric test is the estimation of effects (Sec. 7.2.7, pg. 130). We compare the obtained size of effects against the Cohen's table (provided in Section A.3.2.4, pg. 264) and across the study. The second case study compares three methods of evaluation for the complete problem set using Friedman's rank and Kruskal-Wallis non-parametric tests. The goal is to determine the best combination of qualitative parameters and contribute to structural tuning of BCOc. In Appendix A.3.1.5 we address subject of the data samples for the non-parametric tests in greater detail (pg. 260).

The two case studies are conducted separately for two stopping criteria. Therefore, we distinguish four case studies in total.

8.6 Stopping criterion: N_{it}

In this section we continue to examine the general behavior of BCOc under the maximum number of the iterations. We follow two lines of questioning and define the corresponding hypothesis of our research goals. The first study investigates if the best means reported at different levels of factor Lp ($\bar{y}_{i,j,k,l}$, $\forall j \in \mathcal{L}$ and fixed i, k, l), are significantly different for a particular problem instance. We address this question at each level of parameter ME and define our null hypothesis as follows.

H_0: Loyalty functions produce the same average quality of solutions.

H_1: Loyalty functions produce different average quality of solutions.

We employ the parametric RMANOVA to determine if the null hypothesis holds and non-parametric Friedman's test to validate the results of the parametric test. The samples contain outliers (see Section A.3.1.2, pg 259). Moreover, data samples are paired

(Section A.3.1.3, pg. 260). The condition of sphericity, essential for RMANOVA, is investigated as an integral part of `ez` `R` package [Law15]. However, before employing RMANOVA we examine the conditions of normality following the procedure described in Section A.3.2.1.

The second study of this section is founded on the comparison between methods of evaluation. The null hypothesis is:

H_0: Methods of evaluation have no significant effect on the best average solution quality of the BCOc algorithm.

H_1: Methods of evaluation have significant effect on the best average solution quality of the BCOc algorithm.

Non-parametric statical tests are employed on samples that produced response values $\bar{y}_{i,j}$. In other words, the effect of parameter ME is inspected on the complete problem set and in the design points marked with orange color in the basic tables. Because performance of the BCOc algorithm varies w.r.t. problem specifications, we use ranking procedure. This is addressed in Section A.3.3 (pg. 264).

8.6.1 Case study: the best loyalty function

Graphics of the best mean values from basic tables in Fig. 8.3 demonstrate influence of three factors: Lp, ME and the problem size (n). However, observing the figure the *dominance* of one algorithm against the other is not obvious. To constitute a fair comparison and distinguish the best performing BCOc instance from other the dominance is established by observing the statistical difference and size of effects (η_g^2). The topic of dominance utilized in thesis is elaborated in Section A.3.1.4 (pg. 260). The test of significant differences between the means is conducted at the level of significance $\alpha = 0.05$ and for the sample size $n = n_{run} = 100$.

The results of two RMANOVA and Friedman's tests are presented in Table 8.10. The p-values determine if we reject the null hypothesis of equivalence of means. Table 8.10 is organized along main groups defined by parameter ME. The p-values of the RMANOVA test are shown under column p and of Friedman's test under column $p[f]$. The size of effect of each test is provided in column η_g^2. The first observation is that groups differ, i.e., p-values indicate significant impact of ME. We summarize our observations in Table 8.11.

Firstly, in Table 8.11 we emphasize problem instances for which we fail to reject the null hypothesis for the equivalence of means. We denote with an asterix cases for which the corresponding η_g^2 is the smallest and with double asterix cases for which η_g^2 is the largest. Compared against Cohen's table the largest effect sizes (η_g^2) are exhibited for $\min(ev_1)$, which encompass values from the interval $[0.13, 0.58]$. The large effect size indicates a practical difference, which we address in the following section. Despite significant difference (small p-values), the parameter Lp exhibits small effects ($\eta_g^2 \in [0.02, 0.05]$) in groups $\max(ev_1)$ and $\max(ev_2)$. This indicates a weak practical difference considering that the differences are less then half of unit of time. In particular for $\max(ev_2)$, statistical differences are found in a small set of problem instances.

Results of the RMANOVA and Friedman's tests produce similar p-values and, therefore, similar conclusions regarding influence of parameter Lp. The results of the tests

Table 8.10: Repeated-measure ANOVA and Friedman's test results for equivalence of means between loyalty functions of specific method of evaluation for $\alpha = .05$ and maximum number of iterations.

	$\min(ev_1)$							$\max(ev_1)$							$\max(ev_2)$						
Problem	$p[sph]$	$\hat{\epsilon}$	F	p	η_p^2	η_g^2	$p[f]$	$p[sph]$	$\hat{\epsilon}$	F	p	η_p^2	η_g^2	$p[f]$	$p[sph]$	$\hat{\epsilon}$	F	p	η_p^2	η_g^2	$p[f]$
Iogra100_12	0.000	0.84[2]	24.00	0.000[2]	0.20	0.14	0.000	0.297	0.90	3.96	0.000	0.04	0.03	0.000	0.433	0.91	5.82	0.000	0.06	0.05	0.000
16	0.000	0.59[1]	146.28	0.000[1]	0.60	0.58	0.000	0.605	0.91	2.36	0.012	0.02	0.02	0.007	0.859	0.93	1.06	0.393	0.01	0.01	0.383
Iogra150_12	0.000	0.70[1]	34.62	0.000[1]	0.26	0.19	0.000	0.649	0.92	2.01	0.036	0.02	0.02	0.014	0.139	0.89	1.23	0.273	0.01	0.01	0.254
16	0.000	0.69[1]	52.02	0.000[1]	0.34	0.32	0.000	0.516	0.92	2.34	0.013	0.02	0.02	0.077[5]	0.631	0.92	1.20	0.288	0.01	0.01	0.104
Iogra200_12	0.029	0.97[2]	126.03	0.000[2]	0.56	0.52	0.000	0.783	0.93	1.97	0.039	0.02	0.02	0.033	0.610	0.91	1.15	0.328	0.01	0.01	0.312
16	-	1.00[3]	0.44	0.508[3]	0.00	0.00	0.439	0.845	0.93	2.78	0.003	0.03	0.02	0.009	0.694	0.92	1.48	0.152	0.01	0.01	0.239
Iogra250_12	-	1.00[3]	1.18	0.281[3]	0.01	0.01	0.297	0.062	0.88	2.73	0.004	0.03	0.02	0.003	0.668	0.93	0.46	0.904	0.00	0.00	0.852
16	0.000	0.93[2]	44.08	0.000[2]	0.31	0.27	0.000	0.289	0.90	1.80	0.065	0.02	0.02	0.068	0.254	0.90	1.67	0.093	0.02	0.01	0.242
Iogra300_12	0.008	0.95[2]	0.51	0.720[2]	0.01	0.00	0.599	0.499	0.91	6.23	0.000	0.06	0.05	0.000	0.846	0.93	0.51	0.866	0.01	0.00	0.872
16	0.000	0.91[2]	17.10	0.000[2]	0.15	0.13	0.000	0.553	0.91	1.63	0.103	0.02	0.01	0.131	0.020	0.96[2]	1.24	0.271[2]	0.01	0.01	0.230
Iogra350_12	-	1.00[3]	0.69	0.407[3]	0.01	0.00	0.453	0.895	0.93	6.27	0.000	0.06	0.05	0.000	0.066	0.88	3.69	0.000	0.04	0.03	0.001
16	-	1.00[3]	1.91	0.170[3]	0.02	0.01	0.138	0.996	0.95	1.72	0.081	0.02	0.02	0.102	0.551	0.91	1.84	0.058	0.02	0.02	0.040[6]
Iogra400_12				—				0.687	0.92	3.29	0.001	0.03	0.03	0.001	0.722	0.92	1.03	0.416	0.01	0.01	0.567
16	0.000	0.62[1]	1.09	0.313[1]	0.01	0.01	0.001[6]	0.620	0.92	2.09	0.028	0.02	0.02	0.068[5]	0.884	0.95	2.92	0.005	0.03	0.02	0.005
Iogra450_12	-	1.00[3]	0.34	0.561[3]	0.00	0.00	0.686	0.169	0.90	4.67	0.000	0.05	0.04	0.000	0.931	0.93	2.19	0.021	0.02	0.02	0.042
16	-	1.00[3]	0.15	0.702[3]	0.00	0.00	0.714	0.114	0.88	5.89	0.000	0.06	0.05	0.000	0.913	0.93	6.09	0.000	0.06	0.05	0.000
Iogra500_12	0.000	0.80[2]	86.62	0.000[2]	0.47	0.37	0.000	0.137	0.90	2.54	0.007	0.02	0.02	0.005	0.019	0.95[2]	1.56	0.137[2]	0.02	0.01	0.192
16				—				0.043	0.95[2]	2.96	0.005[2]	0.03	0.03	0.002	0.836	0.97	2.69	0.031	0.03	0.02	0.043

$p[sph]$ - significance (p-value) of Mauchly's statistic test of sphericity;
$\hat{\epsilon}$ - Greenhouse-Geisser estimator of sphericity;
$p[f]$ - significance (p-value) of Friedman's test;

[1] When sphericity condition is violated ($p[sph] < 0.05$) proposed Greenhouse-Geisser estimator is used for correcting p-value of repeated-measures ANOVA test [Dav02];
[2] When sphericity condition is violated ($p[sph] < 0.05$) and $\hat{\epsilon} > 0.75$, Huynh and Feldt estimator $\tilde{\epsilon}$ is used for correcting p-value of repeated-measures ANOVA test [Dav02], p. 111.
[3] '-' When only two data groups differ sphericity condition is always satisfied.
[4] '—' signifies that all outcomes were identical.
[5] Friedman's test failed to reject null hypothesis of equal means for $\alpha = 0.05$. However, repeated-measures ANOVA reported significant effect of Lp on solution quality.
[6] Repeated-measures ANOVA failed to reject null hypothesis of equality of means for $\alpha = 0.05$. Friedman's test reported significant effect of Lp on solution quality.

Table 8.11: Results of RMANOVA and Friedman's test for N_{it}.

ME	class	Significant effect of Lp $p < 0.05$	Range of η_g^2
$\min(ev_1)$	$m = 12$	$n \in \{100^*, 150, 200^{**}, 500\}$.	0.14^*–0.52^{**}
	$m = 16$	$n \in \{100^{**}, 150, 250, 300^*, 400^{(6)}\}$.	0.13^*–0.58^{**}
$\max(ev_1)$	$m = 12$	$n \in \{100, \ldots, 500\}$	0.02–0.05
	$m = 16$	$n \in \{100, 150^{(5)}, 200, 400^{(5)}, 450, 500\}$.	0.02–0.05
$\max(ev_2)$	$m = 12$	$n \in \{100, 350, 450\}$.	0.02–0.05
	$m = 16$	$n \in \{350^{(6)}, 400, 450, 500\}$.	0.02

[5] Friedman's test failed to reject null hypothesis of equal means for $\alpha = 0.05$. However, repeated-measures ANOVA reported significant effect of Lp on solution quality.

[6] Repeated-measures ANOVA failed to reject null hypothesis of equality of means for $\alpha = 0.05$. Friedman's test reported significant effect of Lp on solution quality.

oppose on two problem instance within group $\max(ev_1)$ (marked with (5)). Namely, the Friedman's test did not find enough evidence to distinguish selection of loyalty function based on the best reported values, while RMANOVA did. In cases marked with (6) RMANOVA failed to reject null hypothesis of equivalence of means. Further analysis was conducted by Mk1 test of sample distributions (e.g. A.7) and we confirm RMANOVA results.

8.6.1.1 Post-hoc test for RMANOVA

In previous section we have confirmed that RMANOVA provides valid results for our hypothesis testing. The test determines if there was significant effect of the parameter Lp, however, we still do not know which loyalty function caused this. Therefore, we conduct a pairwise test to identify loyalty functions that caused the significant effects reported in Table 8.10. Contrary to previous analysis, where only samples at levels of Lp were compared, the post-hoc test includes a sample generated by the BCOc reference case (B=20,NC=1).

Because we conduct a controlled study, the multiple comparison procedure is based on the Dunnett's test [Dun55]. The test requires setting a *control group*, which in our study, corresponds to the sample that produced the smallest response value (mean of solutions' quality). Dunnett's test is suitable for reducing the number of pairwise comparisons and avoids an increase of the Type I error. In addition, the test allows comparison between the group with the smallest mean against other groups[(6)]. The results of the pairwise comparison are presented in Table 8.12 for each class of problem instances. The expression $p \geq 0.05$ implies that BCOc parameters have achieved numerical improvements, however, we fail to reject the null hypothesis of Dunnett's test at $\alpha = 0.05$. The most adverse scenario is the one where all the samples are identical. It suggests that performance of all configurations of BCOc parameters are worse than of the `sLPT+BF` heuristic.

[(6)] Most appropriate would be Hsu's multiple comparison test [Hsu96], however, it was not a part of `R` package.

Table 8.12: Results of pairwise comparison test ($N_{it} = 100$): list of loyalty functions that are not significantly different from the control group.

Problem **m=12**, n	$\min(ev_1)$	$\max(ev_1)$	$\max(ev_2)$
100	**8**	$\{\mathbf{0}, 1, 4, 5, 6, 7, 8, 9\}$	$\{0, 1, 2, \mathbf{4}, 5, 6, 7, 8, 9\}$
150	**8**	$\{\mathbf{0}, 1, 2, 4, 5, 6, 7, 8, 9\}$	$\{0, 1, 2, 3, \mathbf{4}, 5, 6, 7, 8, 9\}$
200	$\{\mathbf{2}, 8, 9\}$	$\{0, 1, \mathbf{4}, 5, 6, 7, 8, 9\}$	$\{\mathbf{0}, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
250	$p \geq 0.05$	$\{0, 1, \mathbf{2}, 4, 5, 6, 7, 8, 9\}$	$p \geq 0.05$
300	$p \geq 0.05$	$\{1, 2, 4, 5, \mathbf{8}, 9\}$	$p \geq 0.05$
350	$p \geq 0.05$	$\{0, 1, 4, 6, \mathbf{7}\}$	$\{0, 4, 6, \mathbf{7}\}$
400	—	$\{0, 1, 4, 5, \mathbf{6}, 7, 8, 9\}$	$p \geq 0.05$
450	$p \geq 0.05$	$\{0, 1, 4, 5, \mathbf{6}, 7, 9\}$	$\{\mathbf{0}, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
500	**8**	$\{0, 1, \mathbf{2}, 4, 5, 6, 7, 8, 9\}$	$p \geq 0.05$
Problem **m=16**, n	$\min(ev_1)$	$\max(ev_1)$	$\max(ev_2)$
100	$\{1, \mathbf{2}, 5, 8, 9\}$	$\{\mathbf{0}, 1, 2, 4, 6, 7\}$	$p \geq 0.05$
150	**8**	$\{\mathbf{0}, 1, 4, 5, 6, 7, 8, 9\}$	$p \geq 0.05$
200	$p \geq 0.05$	$\{\mathbf{0}, 1, 4, 5, 6, 7\}$	$p \geq 0.05$
250	$\{2, \mathbf{8}\}$	$p \geq 0.05$	$p \geq 0.05$
300	$\{1, \mathbf{2}, 5, 8, 9\}$	$p \geq 0.05$	$p \geq 0.05$
350	$p \geq 0.05$	$p \geq 0.05$	$p \geq 0.05$
400	$p \geq 0.05$	$\{0, 1, 3, 4, 5, 6, \mathbf{7}, 9\}$	$\{\mathbf{0}, 6, 7, 9\}$
450	$p \geq 0.05$	$\{\mathbf{0}, 6, 7\}$	$\{\mathbf{0}, 3, 4, 7\}$
500	—	$\{\mathbf{0}, 3, 4, 5, 6, 7\}$	$\{0, 3, \mathbf{7}\}$

(1.) $p \geq 0.05$ - Fail to reject Dunnett's test null hypothesis of equivalence of means at $\alpha = 0.05$.
(2.) — All samples are equal and identical with the reference case ($B = 20$, $NC = 1$).

In Table 8.10 at the level $\min(ev_1)$ parameter Lp shows significant effect for problem instances $m = 12 : n \in \{100, 150, 500\}$ and $m = 16 : n = 150$. The pairwise comparison test determines that p^8 is the reason. In addition, for Iogra250_16 the Dunnett's test fails to distinguish its effect from p^2. For the other two problem instances p^2 appears as the control group, however, the test fails to differentiate its success from several others (see Table 8.12).

Within group $\max(ev_1)$ Table 8.12 shows that BCOc achieved better performance than the heuristic at different levels of Lp. However, statistical difference between the corresponding response values, indicated by Table 8.10, are small. Therefore, it is hard to distinguish one loyalty function as the most successful on the complete problem set. For all problem instances mostly the Class II loyalty functions appear as a control group. Compared to others the $p^{0,u}$ function is the most frequent. The p_b^3 function generates the worst overall results since in Table 8.12 appears only two times.

Under influence of the $\max(ev_2)$ evaluation method, other configurations of BCOc parameters essentially do not generate improvements against the heuristic. In particular, both extreme cases are present: $p \geq 0.05$ in Table 8.12 and small size of the effect in Table 8.10. When summarized, the results demonstrate negligible practical effects of Lp. It is worth noting that values of Lp, that have exhibit good results, belong to Class II.

The statistical tests show that the Lp parameter exhibits different effects on the generation of best means. For $N_{it} = 100$ the loyalty functions of class I (especially p^2 and p^8) are the most successful at the level $\min(ev_1)$ w.r.t the heuristic and class II loyalty functions. Both demonstrate good performance w.r.t. quality of the solution and

the running time. Contrary, functions of class II are more successful at levels $\max(ev_1)$ and $\max(ev_2)$. In particular, p_b^0 and p_b^7 yield high quality results compared to the rest Class II functions. When we include results of running times, provided in basic tables, the function p_b^0 demonstrates better performance against p_b^7. However, it exhibits small effects and results of low practical value.

8.6.2 Case study: the effect of ME

In this section we compare three BCOc on the complete benchmark set of problem instances. To determine statistical difference between values of the ME parameter, we use ranking. Since only three methods of evaluation were compared ranks from $[1, 3]$ are used. Computation of ranks for each problem instance is done by procedures of multiple comparison using RMANOVA to test the hypothesis of equivalence of means, and pairwise t-test with Hommel's correction as a post-hoc test to resolve ties. We address this in Section A.3.3 with detailed description of the implementation with R (pg. 264). Results of these tests are in great part omitted because of overwhelming number of comparisons. However, we try to provide as much information as possible in tables of ranks.

Friedman's and Kruska-Wallis non-parametric tests are utilized on the ranks. The null hypothesis of the Friedman's and Kruskal-Wallis tests is that samples come from the same population[(7)]. When no significant difference is detected, groups of ME are evaluated by average values of the corresponding relative errors and ranks. To distinguish contributions of BCOc configurations, preliminary analysis showed that non-parametric tests should be employed independently on two classes of problem instances. The results are organized in Tables 8.13 and 8.14.

The structure of Tables 8.13 and 8.14 is as follows. Column $r.err.$ designates relative errors of the corresponding best means. Column r represents ranks determined at each level of problem class. Column $\bar{y}$ shows the overall best result and its deviation of the corresponding method of evaluation. The average running times are given in column $\bar{t}$. The best configuration of quantitative BCOc parameters together with the corresponding loyalty function are provided under columns B, NC and p_b. Average values of relative errors and ranks are placed at the bottom of a table. They provide the first insight about the overall performance of the considered BCOc evaluation methods. To avoid ambiguity we use symbol (1) when ever statistical tests, failed to reject null hypothesis of equivalence of means at the 5% level of significance.

The results of two non-parametric tests conducted on ranks in Tables 8.13 and 8.14 are as follows.

Table 8.13: (a) With Friedman's test we fail to reveal significant effect of ME on reporting of the best solution quality ($\chi^2 = 2.25$, $p = 0.325$). (b) With Kruskal-Wallis test we fail to reveal significant effect of ME on reporting of the best solution quality ($\chi^2 = 2.99$, $p = 0.224$).

Table 8.14: (a) With Friedman's test we fail to reveal significant effect of ME on reporting of the best solution quality ($\chi^2 = 0.143$, $p = 0.71$). (b) With Kruskal-

(7) The null hypothesis may also include testing equivalence of medians. However, when using Kruskal-Wallis to test equivalence of medians, samples should come from a same distribution [McD14, pp. 157-164].

Table 8.13: Table of best results and its corresponding parameter configurations for class of $m = 12$ of problem instances. Stopping criterion, $N_{it} = 100$.

	$\min(ev_1)$							$\max(ev_1)$							$\max(ev_2)$						
Probl. inst.	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	*B*	*NC*	p_b	r	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	*B*	*NC*	p_b	r	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	*B*	*NC*	p_b	r
100	810.29 ± 3.54	1.29	45.10 ± 12.00	20	97	p_b^8	**1**	812.58 ± 1.34	1.57	35.90 ± 9.99	20	96	p_b^0	**3**	812.07 ± 1.63	1.51	33.70 ± 5.87	20	47	p_b^4	**2**
150	1003.54 ± 1.25	0.35	58.30 ± 16.50	20	74	p_b^8	**1**	1004.85 ± 0.70	0.49	42.30 ± 13.80	20	71	p_b^0	**2.5**	1004.99 ± 0.82	0.50	29.50 ± 9.09	20	7	p_b^4	**2.5**
200	1202.75 ± 0.92	0.23	59.80 ± 22.30	19	91	p_b^2	**1**	1204.32 ± 0.61	0.36	50.50 ± 11.30	20	22	p_b^4	**2**	1204.51 ± 0.62	0.38	45.60 ± 16.60	20	39	p_b^0 (1)	**3**
250	1404.07 ± 0.57	0.29	46.10 ± 21.50	20	2	p_b^3 (1)	**2.5**	1403.65 ± 0.55	0.26	71.80 ± 12.90	19	41	p_b^2	**1**	1403.91 ± 0.55	0.28	50.90 ± 20.60	20	12	p_b^0 (1)	**2.5**
300	1603.51 ± 0.95	0.22	83.00 ± 39.20	19	85	p_b^2 (1)	**2.5**	1603.15 ± 0.55	0.20	90.70 ± 17.80	20	40	p_b^8	**1**	1603.52 ± 0.57	0.22	59.70 ± 26.30	20	5	p_b^0 (1)	**2.5**
350	1807.14 ± 0.80	0.40	90.00 ± 29.80	20	2	p_b^3 (1)	**3**	1806.50 ± 0.82	0.36	96.00 ± 28.90	20	7	p_b^7	**1.5**	1806.67 ± 0.81	0.37	89.80 ± 31.70	20	7	p_b^7	**1.5**
400	2004.28 ± 0.69	0.21	103.00 ± 34.00	20	1	p_b^0 (1)	**3**	2003.85 ± 0.62	0.19	132.00 ± 29.60	20	12	p_b^6	**1**	2004.05 ± 0.61	0.20	109.00 ± 33.20	20	7	p_b^7 (1)	**2**
450	2205.02 ± 0.62	0.23	143.00 ± 39.70	20	9	p_b^3 (1)	**3**	2204.48 ± 0.64	0.20	149.00 ± 39.30	20	7	p_b^6	**1**	2204.69 ± 0.70	0.21	159.00 ± 50.60	20	83	p_b^0 (1)	**2**
500	2402.58 ± 0.99	0.11	239.00 ± 58.00	20	96	p_b^8	**1**	2403.36 ± 0.71	0.14	203.00 ± 30.60	19	63	p_b^2	**2**	2403.62 ± 0.54	0.15	138.00 ± 52.80	20	9	p_b^7 (1)	**3**
Av.	1604.80	0.37						1605.19	0.42						1605.34	0.42					

(1) Differences between best results reported by loyalty fuctions $p^i, i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ on this instance are statistically insignificant. Parameters configuration that generated the best numerical value is provided.

Table 8.14: Table of best results and its corresponding parameter configurations for class of $m = 16$ of problem instances. Stopping criterion, $N_{it} = 100$.

	$\min(ev_1)$							$\max(ev_1)$							$\max(ev_2)$						
Probl. inst.	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	*B*	*NC*	p_b	r	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	*B*	*NC*	p_b	r	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	*B*	*NC*	p_b	r
100	805.55 ± 1.41	0.69	43.70 ± 16.10	20	92	p_b^2	**1**	808.77 ± 0.87	1.10	37.90 ± 11.20	19	90	p_b^0	**2.5**	808.68 ± 1.04	1.08	49.80 ± 10.00	20	94	p_b^4	**2.5**
150	1005.69 ± 1.30	0.57	66.20 ± 19.60	20	76	p_b^8	**1**	1008.49 ± 0.91	0.85	42.00 ± 8.09	20	10	p_b^0	**2.5**	1008.42 ± 0.91	0.84	57.40 ± 17.00	20	95	p_b^0	**2.5**
200	1208.26 ± 0.88	0.69	54.00 ± 12.30	20	2	p_b^3 (1)	**3**	1207.91 ± 0.83	0.66	68.80 ± 16.90	19	67	p_b^0	**1.5**	1207.86 ± 0.89	0.65	68.80 ± 17.10	19	67	p_b^0	**1.5**
250	1404.19 ± 1.15	0.30	97.00 ± 33.40	20	90	p_b^8	**1**	1405.83 ± 0.68	0.42	87.10 ± 23.00	20	72	p_b^0 (1)	**2.5**	1405.90 ± 0.71	0.42	88.90 ± 27.00	19	84	p_b^0	**2.5**
300	1605.06 ± 1.13	0.32	114.00 ± 33.50	19	91	p_b^2	**1**	1605.81 ± 0.59	0.36	111.00 ± 28.30	19	90	p_b^0 (1)	**2.5**	1605.86 ± 0.60	0.37	81.00 ± 23.80	19	5	p_b^7	**2.5**
350	1805.72 ± 0.53	0.32	104.00 ± 29.20	20	2	p_b^3	**2**	1805.53 ± 0.62	0.31	137.00 ± 39.60	20	97	p_b^0 (1)	**2**	1805.60 ± 0.69	0.31	130.00 ± 37.30	20	88	p_b^0 (1)	**2**
400	2005.78 ± 2.33	0.29	156.00 ± 32.40	15	99	p_b^8 (1)	**2**	2005.75 ± 0.65	0.29	136.00 ± 34.00	20	12	p_b^7	**2**	2005.70 ± 0.69	0.29	148.00 ± 34.20	20	40	p_b^0 (1)	**2**
450	2207.76 ± 0.74	0.35	173.00 ± 37.40	20	2	p_b^3 (1)	**3**	2207.23 ± 0.76	0.33	204.00 ± 34.60	19	97	p_b^0	**1.5**	2207.18 ± 0.79	0.33	200.00 ± 41.40	20	92	p_b^0	**1.5**
500	2407.35 ± 0.70	0.31	191.00 ± 41.70	20	1	p_b^0 (1)	**3**	2407.03 ± 0.70	0.29	229.00 ± 37.90	20	45	p_b^0	**1.5**	2407.09 ± 0.75	0.30	200.00 ± 41.50	20	11	p_b^7 (1)	**1.5**
Av.	1606.15	0.43						1606.93	0.51						1606.92	0.51					

(1) Differences between best results reported by loyalty fuctions $p^i, i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ on this instance are statistically insignificant. Parameters configuration that generated the best numerical value is provided.

Wallis test we fail to reveal significant effect of ME on reporting of the best solu-

tion quality ($\chi^2 = 0.25$, $p = 0.62$).

The tests do not suggest significant difference among the considered methods of evaluation for both problem classes $m = \{12, 16\}$. The reason why tests have failed to reject the null hypothesis arises from the restrictions of the $\mathcal{NC}$ domain. The ties among values of ME are resolved by comparing average values of the relative errors and ranks over all test instances. We conclude that $\min(ev_1)$ for this particular problem set is the best performing method of evaluation. Methods $\max(ev_1)$ and $\max(ev_2)$ exhibit similar performance comparing across the problem set. When we restrict the observations on the implementation of evaluation function ev_2, no improvements are detected. Moreover, evaluation function ev_2 has degraded the performance most often compared to ev_1.

8.6.2.1 Quantitative parameters, B and NC

We devote this part of the section to investigate values of the quantitative parameters B and NC that contributed to previous conclusions. We observe Tables 8.13 and 8.14. For problem instances $m = 12$ in 95% (of totally 27) the maximal population size ($B = 20$) generates the best results. The rest 5% corresponds to $B = 19$. For class $m = 16$ 68% of the time the best results were generated for $B = 20$. In the rest of the cases the population size is $B = 19$, with an exception of $B = 15$ for $n = 400$.

Regarding the influence of NC on the outcome, different response values are reported. Tables 8.13 and 8.14 show that NC should be somewhere between $[74, 99]$ for parameter configuration $\min(ev_1) : p_b^{2,8}$. For Class II loyalty functions a strong conclusion can be drawn only for p_b^7, since the best reported outcomes are when $NC \in [5, 12]$. Function p_b^0 is influenced by the size of the problem instance as there is no distinctive pattern within values of NC, therefore, ranging between $[1, 97]$.

8.6.2.2 Graphics of comparison between levels of ME

After evaluation of all configurations and their statistical difference we employ graphical representation of the best response values reported in Tables 8.13 and 8.14. The choice of parameters' configuration was extensively elaborated in previous case studies and more information about the best representatives can be found in (Tables 8.13 and 8.14). As the performance measure we use the relative error. In Fig. 8.7 four profile lines are shown for each problem class. Dotted line signifies the *reference quality* determined by solutions of the standalone heuristics `sLPT+BF`. The colored lines correspond to solutions' mean produced by best BCOcs, as indicated in Tables 8.13 and 8.14.

The graphics illustrate improvements of $\min(ev_1)$ against other three methods. Although not distinguishable, for $m = 12 : n \geq 200, n \neq 500$ the method $\max(ev_1)$ provides solutions that are significantly different from results of other two methods of evaluation. The conclusions is founded on ranks in Table 8.13. Contrary, for $m = 16$ the method $\min(ev_1)$ is dominant in $n = \{100, 150, 250, 300\}$, while both maximization evaluation methods generated slightly better results for the rest of problem instances. The graphic also illustrates the size of difference achieved at different levels of ME. Namely, $\min(ev_1)$ improves the performance of the underlying heuristic `sLPT+BF` much more than the maximization strategy. Additionally, the BCOc algorithm with the $\min(ev_1)$ succeeds to find parameter configuration in which the quality is not worse against the `sLPT+BF`.

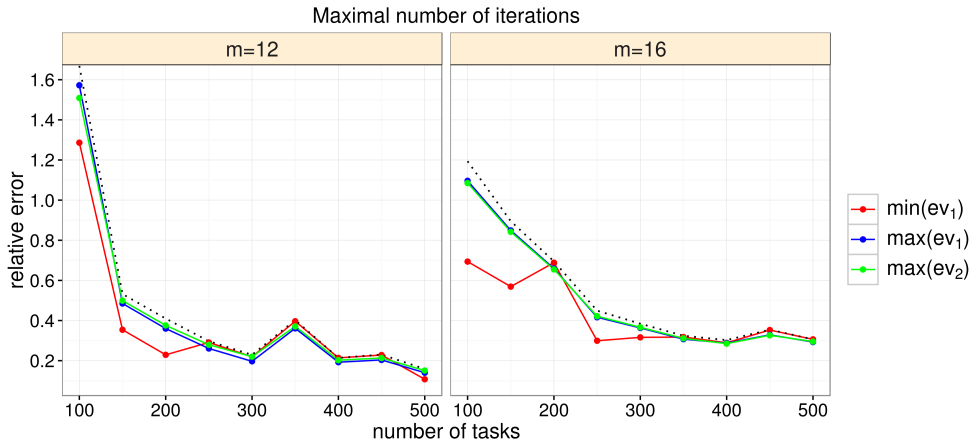


Figure 8.7: Graphic of propagation of best results generated by each method of evaluation when maximal number of iterations is provided. Dotted line corresponds to performance of sLPT+BF.

8.7 Stopping criterion: CPU time

Providing time as a stopping criterion may change the conclusions made previously. However, we restrict CPU time to establish objective environment inside which less care may be taken about the fairness regrading the total running time (thus number of evaluations). It is expected that allowing more time differentiates clearly between impacts of loyalty functions on the solution quality. The results of this study are presented in the similar manner as previously. We skip some of the steps, such as RMANOVA and Friedman's test results for equivalence of means between loyalty functions (Table B.1, pg. 273) and directly provide the conclusions regarding performance of parameter Lp and ME.

8.7.1 Case study: the best loyalty function

Results in Table B.1 reveal significant effect of the parameter Lp on the response values. Observations regarding results of RMANOVA test are summarized in Table 8.15. The table shows that an impact of the $\max(ev_1)$ has increased compared to results of N_{it}, indicating improvements against the standalone heuristic. In particular, Tables 8.15 and B.1 reveal that $\max(ev_1)$ has produced small p-values for each problem instance, contrary to other two methods of evaluation. Occasionally, the RMANOVA and Friedman's test disagree in the same manner as reported in Table 8.11.

Regrading the results of the Dunnett's test (Table 8.16), in group $\min(ev_1)$ the p^2 improved results for problem instance $m = 16 : n \in \{150, 250\}$. The p^8 generates high quality results for Iogra400_16, contrary to N_{it} where neither value of Lp improved results against sLPT+BF. More interesting is appearance of p_b^3 as the control group for the test instances $m = 12 : n = 350$ and $m = 16 : n \in \{350, 450\}$. It achieved better results than the standalone heuristic for cases where no significant difference was detected under N_{it}. However, the corresponding effect size (η_g) reported in Table B.1 is small, indicating small significance in difference between generated samples.

Table 8.15: Results of RMANOVA and Friedman's test for T.

ME	class	Significant effect of Lp $p < 0.05$	Range of η_g^2
$\min(ev_1)$	$m = 12$	$n \in \{100, 150, 200^{**}, 300^{(6)}, 350^{*}, 500\}$.	0.03^{*}–0.53^{**}
	$m = 16$	$n \in \{100^{**}, 150, 200, 250, 300, 350, 400, 450^{*}\}$.	0.02^{*}–0.57^{**}
$\max(ev_1)$	$m = 12$	$n \in [100, \ldots, 500]$.	0.15–0.36
	$m = 16$	$n \in [100, \ldots, 500]$.	0.06–0.24
$\max(ev_2)$	$m = 12$	$n \in \{100, 150^{(5)}, 350\}$.	0.02–0.20
	$m = 16$	$n \in \{100, 150, 200, 250, 300^{(5)}, 450\}$.	0.02–0.07

[5] Friedman's test failed to reject null hypothesis of equal means for $\alpha = 0.05$. However, repeated-measures ANOVA reported significant effect of Lp on solution quality.
[6] Repeated-measures ANOVA failed to reject null hypothesis of equality of means for $\alpha = 0.05$. Friedman's test reported significant effect of Lp on solution quality.

In group $\max(ev_1)$ we observe that p_b^2 appears frequently as the control group. Coupled with the corresponding results in Table B.1, we may conclude that the statistical difference between the dominant collection of loyalty functions and the non-dominant is significant.

In group $\max(ev_2)$, the results indicate that Iogra100_12 is the only instance for which parameter Lp demonstrates any significant impact. For the rest of problem instances, difference in the performance is practically important. This concludes that $\max(ev_2)$ should not be employed within the BCOc algorithm.

Table 8.16: Results of pairwise comparison test ($T = n/100[s]$): list of loyalty functions that are not significantly different from the control group.

Problem **m=12**, n	$\min(ev_1)$	$\max(ev_1)$	$\max(ev_2)$
100	$\{2, \mathbf{8}\}$	$\{\mathbf{2}, 8\}$	$\{1, \mathbf{2}, 5, 6, 8\}$
150	$\mathbf{8}$	$\{1, \mathbf{2}, 5, 8, 9\}$	$\{0, 1, \mathbf{2}, 4, 5, 6, 7, 8, 9\}$
200	$\{1, \mathbf{2}, 5, 8\}$	$\{1, \mathbf{2}, 5, 8, 9\}$	$p > 0.05$
250	$p > 0.05$	$\{1, \mathbf{2}, 5, 8, 9\}$	$p > 0.05$
300	$p > 0.05$	$\{1, 2, 5, 6, \mathbf{8}, 9\}$	$p > 0.05$
350	$\mathbf{3}$	$\{1, \mathbf{2}, 5, 8, 9\}$	$\{\mathbf{0}, 1, 2, 4, 5, 6, 7, 8, 9\}$
400	$p > 0.05$	$\{1, \mathbf{2}, 5, 6, 8, 9\}$	$p > 0.05$
450	$p > 0.05$	$\{1, \mathbf{2}, 5, 8, 9\}$	$p > 0.05$
500	$\mathbf{8}$	$\{1, 2, 5, 6, \mathbf{8}, 9\}$	$p > 0.05$
Problem **m=16**, n	$\min(ev_1)$	$\max(ev_1)$	$\max(ev_2)$
100	$\{1, \mathbf{2}, 5, 8\}$	$\{1, \mathbf{2}, 4, 5, 6, 8, 9\}$	$\{\mathbf{2}, 8\}$
150	$\{\mathbf{2}, 8\}$	$\{1, \mathbf{2}, 5, 8, 9\}$	$\{\mathbf{2}, 5, 8\}$
200	$\mathbf{8}$	$\{1, \mathbf{2}, 5, 8\}$	$\{0, 1, \mathbf{2}, 4, 5, 6, 7, 8, 9\}$
250	$\{\mathbf{2}, 8\}$	$\{1, 2, 5, 6, \mathbf{8}, 9\}$	$\{1, \mathbf{2}, 5, 8\}$
300	$\{1, \mathbf{2}, 5, 8\}$	$\{1, \mathbf{2}, 5, 6, 8, 9\}$	$\{0, 1, \mathbf{2}, 4, 5, 6, 7, 8\}$
350	$\mathbf{3}$	$\{1, \mathbf{2}, 4, 5, 6, 8, 9\}$	$p > 0.05$
400	$\mathbf{8}$	$\{1, \mathbf{2}, 5, 6, 8, 9\}$	$p > 0.05$
450	$\mathbf{3}$	$\{1, \mathbf{2}, 5, 8, 9\}$	$\{0, 1, 2, 4, 5, \mathbf{6}, 7, 8, 9\}$
500	$p > 0.05$	$\{1, \mathbf{2}, 5, 8, 9\}$	$p > 0.05$

[1] $p > 0.05$ - With one-way repeated measures ANOVA we failed to find significant effect of loyalty function on the solution quality.

8.7.2 Case study: the effect of ME

The results found in basic tables are grouped in Tables 8.17 and 8.18. All ranks are marked as bold, determined at each level of a problem instance (Section 8.6.2). The results of two non-parametric tests conducted on ranks in Tables 8.17 and 8.18 are as follows.

Table 8.13: (a) With Friedman's test we fail to reveal significant effect of ME on reporting of the best solution quality ($\chi^2 = 5.88$, $p = 0.053$). (b) Kruskal-Wallis test reveals significant effect of ME on reporting of the best solution quality ($\chi^2 = 0.62$, $p = 0.013$).

Table 8.14: (a) Friedman's test reveals significant effect of ME on reporting of the best solution quality ($\chi^2 = 6.2$, $p = 0.045$). (b) Kruskal-Wallis test reveals significant effect of ME on reporting of the best solution quality ($\chi^2 = 8.77$, $p = 0.001$).

The tests suggest significant effect of the parameter ME. The Friedman's test for $m = 12$ produced p-value close to 0.05 while the Kruskal-Wallis produced small p-value. For $m = 16$ both tests demonstrate significance of difference in performance of BCOcs. Contrary to the previous case study, the dominance of a particular loyalty function is here clearer established. For $m = 12$ most successful is p_b^2 with occurring ∼37% (10 times) in Table 8.17. Another successful functions is p_b^8 with 8 times, p_b^3 with 4 times, p_b^7 with 2 times, p_b^0 with 1 time. For $m = 16$ and in regard to loyalty functions, most successful is p_b^2 with occurring ∼63% (17 times) in Table 8.17. Another successful functions were p_b^8 with 4 times, p_b^3 with 2 times, p_b^7 with 2 times, p_b^6 with 2 times.

8.7.2.1 Quantitative parameters, B and NC

The *large* (maximal) population size is considered to be all values of $B \geq 14$, the *medium* size when $7 \leq B \leq 13$, and *small* size when $B \leq 6$. For $m = 12$ the maximal population size generates the best results in only 18.5% (5 cases of totally 27). The majority of high quality results is reported for both small and medium populations, i.e., each appear ∼ 40.7% (11 times).

For $m = 16$ in 6 cases the maximal population size generates the best results in ∼ 26% (7 times), all within $\min(ev_1)$. The medium population size is mostly detected for $\max(ev_2)$, that is in ∼70% within the group and 22% totally. Most frequent is the small population size with ∼48% occurrences, and 100% within the group $\max(ev_2)$.

The evaluation of the NC parameter is deduced by observing results of BCOs with ranks 1. Therefore, only results of two methods of evaluation were considered: $\min(ev_1)$ and $\max(ev_1)$. For $\min(ev_1)$ the values appeared in interval $[72, 96]$. For $\max(ev_1)$ the NC values appeared in interval $[45, 78]$.

8.7.2.2 Graphics of comparison between levels of ME

In Fig. 8.8 when $m = 12$ all algorithms show similar similar tendencies in average. For example, when $n > 200$, $\max(ev_1)$ is dominant over the two BCOs, $\min(ev_1)$ and $\max(ev_2)$ which exibit similar performances. However, for $n \leq 200$ the method $\min(ev_1)$ performs much better than the other two. A more straightforward conclusion can be drawn if $m = 16$, as $\min(ev_1)$ performed the best on majority of the test

Table 8.17: Table of best results and its corresponding parameter configurations for class of $m = 12$ of problem instances. Stopping criterion: CPUtime.

	$\min(ev_1)$							$\max(ev_1)$							$\max(ev_2)$						
Probl. inst.	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	B	NC	p_b	r	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	B	NC	p_b	r	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	B	NC	p_b	r
100	807.61 ± 2.50	0.95	64.40 ± 21.30	16	97	p_b^8	**1**	808.47 ± 0.68	1.06	72.50 ± 18.80	7	25	p_b^2	**2**	808.85 ± 1.06	1.11	63.70 ± 22.60	11	9	p_b^2	**3**
150	1002.38 ± 0.82	0.24	90.50 ± 36.30	20	72	p_b^8	**1**	1003.12 ± 0.52	0.31	91.00 ± 37.50	10	37	p_b^2	**2**	1003.71 ± 0.60	0.37	78.70 ± 37.70	2	4	p_b^2	**3**
200	1202.27 ± 0.73	0.19	86.60 ± 45.90	20	91	p_b^2	**1**	1202.93 ± 0.43	0.24	97.30 ± 50.40	8	34	p_b^2	**2**	1203.54 ± 0.56	0.29	88.30 ± 52.10	6	5	p_b^8 (1)	**3**
250	1403.28 ± 0.49	0.23	99.30 ± 65.90	3	11	p_b^3 (1)	**2.5**	1402.59 ± 0.49	0.18	104.00 ± 55.50	11	50	p_b^2	**1**	1403.15 ± 0.52	0.23	103.00 ± 66.40	4	6	p_b^6 (1)	**2.5**
300	1602.81 ± 0.80	0.18	141.00 ± 84.40	16	70	p_b^8 (1)	**2.5**	1602.39 ± 0.49	0.15	142.00 ± 71.20	11	45	p_b^8	**1**	1602.91 ± 0.51	0.18	131.00 ± 77.30	12	7	p_b^4 (1)	**2.5**
350	1805.86 ± 0.76	0.33	209.00 ± 80.00	6	4	p_b^3	**3**	1804.27 ± 0.60	0.24	216.00 ± 69.90	10	55	p_b^2	**1**	1805.53 ± 0.68	0.31	181.00 ± 86.60	5	8	p_b^0	**2**
400	2003.47 ± 0.61	0.17	178.00 ± 103.00	2	6	p_b^3 (1)	**3**	2002.62 ± 0.54	0.13	198.00 ± 83.60	11	51	p_b^2	**1**	2003.18 ± 0.54	0.16	203.00 ± 93.40	3	5	p_b^8 (1)	**2**
450	2204.10 ± 0.64	0.19	245.00 ± 115.00	3	2	p_b^3	**3**	2203.00 ± 0.55	0.14	265.00 ± 97.10	12	78	p_b^2	**1**	2203.81 ± 0.58	0.17	227.00 ± 105.00	6	7	p_b^7 (1)	**2**
500	2401.71 ± 0.62	0.07	339.00 ± 110.00	20	96	p_b^8	**1**	2402.43 ± 0.50	0.10	264.00 ± 113.00	12	67	p_b^8	**2**	2403.05 ± 0.50	0.13	216.00 ± 135.00	3	8	p_b^7 (1)	**3**
Av.	1603.72	0.28						1603.54	0.28						1604.19	0.33					

(1) Differences between best results reported by loyalty fuctions $p^i, i \in \{0,1,2,3,4,5,6,7,8,9\}$ on this instance are statistically insignificant. Parameters configuration that generated the best numerical value is provided.

Table 8.18: Table of best results for $m = 16$.

	$\min(ev_1)$							$\max(ev_1)$							$\max(ev_2)$						
Probl. inst.	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	B	NC	p_b	r	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	B	NC	p_b	r	$\bar{y}$ (s.d.)	*r.err.*	$\dot{t}$ (s.d.) $[10^{-3}]$	B	NC	p_b	r
100	804.97 ± 1.23	0.62	54.50 ± 22.80	14	96	p_b^2	**1**	806.83 ± 0.63	0.85	66.20 ± 22.40	6	6	p_b^2	**3**	806.64 ± 0.71	0.83	60.60 ± 22.90	2	82	p_b^2	**2**
150	1004.86 ± 1.17	0.49	86.70 ± 28.70	19	77	p_b^2	**1**	1006.40 ± 0.63	0.64	89.00 ± 32.00	5	37	p_b^2	**2.5**	1006.35 ± 0.86	0.64	99.10 ± 30.00	3	67	p_b^2	**2.5**
200	1205.02 ± 1.57	0.42	147.00 ± 36.70	19	100	p_b^8	**1**	1205.72 ± 0.63	0.48	132.00 ± 39.40	7	38	p_b^2	**2**	1206.27 ± 0.65	0.52	112.00 ± 47.10	4	5	p_b^2	**3**
250	1403.61 ± 1.02	0.26	136.00 ± 57.40	19	98	p_b^2	**1**	1404.42 ± 0.57	0.32	142.00 ± 52.60	7	25	p_b^8	**2.5**	1404.54 ± 0.57	0.32	145.00 ± 53.90	4	75	p_b^2	**2.5**
300	1604.31 ± 0.86	0.27	163.00 ± 70.60	18	92	p_b^2	**1.5**	1604.44 ± 0.54	0.28	167.00 ± 69.90	8	44	p_b^2	**1.5**	1604.84 ± 0.63	0.30	175.00 ± 66.80	4	98	p_b^2	**3**
350	1804.91 ± 0.58	0.27	167.00 ± 94.40	3	3	p_b^3	**2.5**	1804.29 ± 0.48	0.24	195.00 ± 76.00	7	51	p_b^2	**1**	1804.83 ± 0.47	0.27	167.00 ± 86.80	3	9	p_b^7 (1)	**2.5**
400	2003.22 ± 1.02	0.16	265.00 ± 77.80	20	100	p_b^8	**1**	2004.33 ± 0.53	0.22	246.00 ± 82.50	8	67	p_b^2	**2**	2004.90 ± 0.56	0.25	213.00 ± 92.90	6	9	p_b^6 (1)	**3**
450	2206.31 ± 0.76	0.29	265.00 ± 105.00	2	33	p_b^3 (1)	**3**	2205.10 ± 0.57	0.23	290.00 ± 92.40	6	71	p_b^2	**1**	2205.90 ± 0.78	0.27	267.00 ± 100.00	3	10	p_b^6	**2**
500	2406.11 ± 1.58	0.25	375.00 ± 86.60	15	98	p_b^8 (1)	**2.5**	2405.25 ± 0.57	0.22	345.00 ± 95.40	7	56	p_b^2	**1**	2406.00 ± 0.73	0.25	276.00 ± 124.00	3	10	p_b^7 (1)	**2.5**
Av.	1604.81	0.34						1605.20	0.39						1605.59	0.41					

(1) Differences between best results reported by loyalty fuctions $p^i, i \in \{0,1,2,3,4,5,6,7,8,9\}$ on this instance are statistically insignificant. Parameters configuration that generated the best numerical value is provided.

instances, while performances of $\max(ev_1)$ and $\max(ev_2)$ coincide. The general improvement over sLPT+BF heuristic is statistically and practically significant for instances

$m = 12 : n = \{100, 150, 200\}$ and $m = 16 : n = \{100, 150, 200, 250, 400\}$

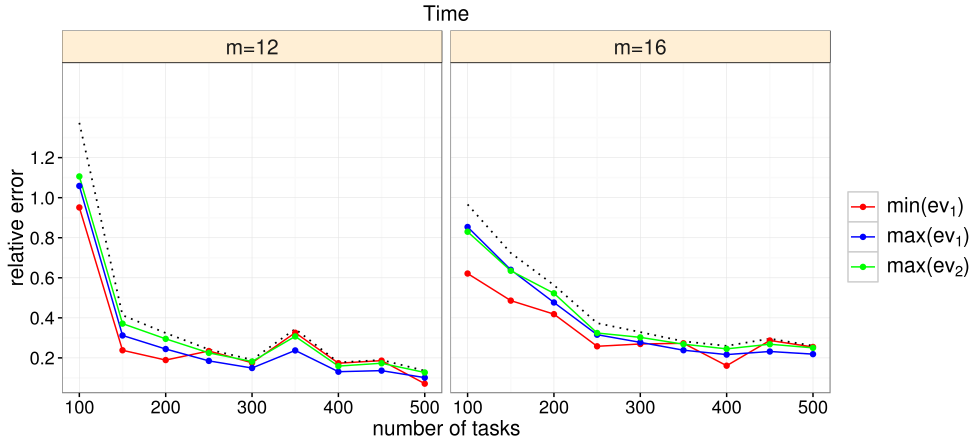


Figure 8.8: Graphic of propagation of best results generated by each method of evaluation when maximal CPU time is provided. Dotted line corresponds to performance of `sLPT+BF`.

8.8 Recruitment dynamics

Before concluding the results of statistical and graphical analysis, the influence of different qualitative parameters of BCOc algorithm may be further analyzed by the dynamics behind the recruitment process. The recruitment dynamics is a process that depends on the number of recruiters, generated during the execution of the BCOc algorithm. We need to consider that the number of recruiters is different in each backward pass and to capture all the necessary outcomes, (for each considered configuration of parameters) is often computationally expensive. To decrease computational cost and to examine the recruitment dynamics all the case studies were carried out for the fixed value of parameter B. In addition, all case studies that concern recruitment process of BCOc utilize the data generated for the stopping criterion controlled by the maximal number of iterations. In order to capture the recruitment dynamics for different configurations of BCO parameters and different *seeds*, we employ a new variable R_e. It represents an average number of recruiters generated over all runs of an experiment, that is, for different values of $seed$. Specifically, values of R_e were generated during series of experiments, each experiment defined by different value of NC, loyalty function and method of evaluation. The experiments reveal enough information about existing tendencies or patterns behind the recruitment process. Before providing results, it should be noted that R_e does not depend on the properties of the test instance but rather on analytical expression of objective and loyalty function. This is confirmed by running tests on more then one problem instance. Therefore, we have conducted the experiments for $m = 12$ and $n = 100$.

To generate values of R_e, we introduce new variables, R_i and $\overline{R}$. The procedure of calculating these variables is as follows. Variable R is usually used to control the backward pass (described in Section 4.2.3), therefore initialized at the beginning of each iteration. As it changes within each backward pass, R can be presented as a

function of forward/backward pass counter, u. Since in each iteration backward pass is called at most NC times, the average number of recruiters within iteration i of BCO is determined as R_i:

$$R_i = \frac{1}{NC} \sum_{u=1}^{NC} R(u), \text{ where } 1 \leq i \leq N_{it}.$$

To further simplify examination of the recruitment process, R_i is averaged over all iterations thus yielding $\overline{R}$ calculated as:

$$\overline{R} = \frac{1}{N_{it}} \sum_{i=1}^{N_{it}} R_i.$$

We expected the influence of $seed$ on values of $\overline{R}$, for the particular problem instance, to show large variability. Nevertheless, the experiment is conducted by repeating BCOc algorithm for different values of $seed$ ($n_{run} = 100$ independent runs), thus generating parameter R_e as:

$$R_e = \frac{1}{n_{run}} \sum_{i=1}^{n_{run}} \overline{R}.$$

The results of the first case study, presented in Fig. 8.9, are grouped by loyalty functions, while within each group we can distinguish values provided by three methods of evaluation. From Fig. 8.9 we observe the following. Firstly, the influence of methods of evaluation on R_e for some cases of loyalty functions is not distinguishable, such as $p_b^{0,3,7}$. In other words, these loyalty functions are robust to utilization of different evalu-

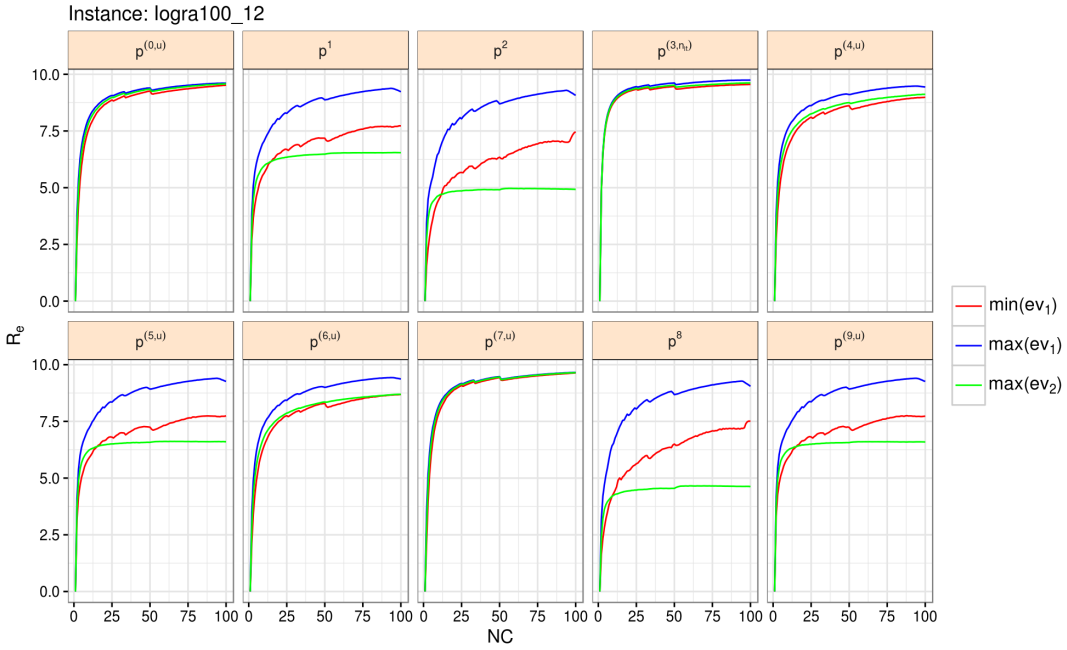


Figure 8.9: Propagation of variable R_e (average number of recruiters per experiment) over $NC \in [1, 100]$ for $B = 10$ and instance *logra100_12*.

ation functions. Possible explanation is that that these three loyalty functions converge fast towards the probability space where is most likely for the partial solution to remain in the population set used in the next forward step. Similarly, functions $p_b^{4,6}$ show tendency to be insensitive to evaluation process. In addition, the two loyalty functions had exhibit similar behavior when $min(ev_1)$ and $max(ev_2)$ are used. Secondly, conclusions are made for each case of method of evaluation. It seems that utilizing $max(ev_1)$ always provides the highest number of recruiters, regardless of the loyalty function. Methods $min(ev_1)$ and $max(ev_2)$, however, are sensitive to the choice of loyalty function. Furthermore, as the NC is increasing, it seems that method $max(ev_2)$ reaches a certain threshold for R_e after which it remains constant. This behavior is very clear for $p_b^i, i \in 1, 2, 5, 8, 9$. For example, case $\{p_b^2, max(ev_2)\}$ shows that after NC reaches 4, the variable R_e remains almost constant (doesn't exceed 5). This does not reflect situation where only 4 recruiters are generated in each step of the iteration, but more that the dynamics of recruitment remains similar on the considered domain of NC. We observe $min(ev_1)$ for $p_b^i, i \in 1, 2, 5, 8, 9$. It seems that this method generates more recruiters per each forward/backward step, however, variable R_e mostly doesn't exceed 7.5. Furthermore, the tests show that the number of recruiters within iteration can range from the smallest (0) to the largest (10).

The conclusions based solely on results in Fig. 8.9 obviously don't provide enough information about what is causing $max(ev_1)$ to generate such high profiles. In order to gather new insights, a new set of tests is conducted. To provide clearer distribution of the number of recruiters during each forward/backward phase, one needs to consider two possible extreme scenarios that can lead to an increase in the recruitment population of bees: (1) satisfiability of condition $C_{min} = C_{max}$, (2) $R = B$. Both cases cause $R = B$, however, if the first condition is satisfied the BCO algorithm skips backward pass completely, whereas the second condition is determined after second phase of the backward pass. The first condition is named *miss* of the forward/backward pass. Therefore, within BCOc algorithm three parameters were followed: R_e, R_{miss}, R_{max}. Specifically, to generate values of R_{miss} and R_{max} reported at the end of an experiment, a counter is set in the BCO instance that increased each time condition $C_{min} = C_{max}$ is satisfied before the start of backward pass, or $R = B$ within the backward pass, respectively. The results of two studies, performed in similar fashion as previously, are presented in Fig. 8.10 and 8.11. Both figures provide more understanding of the performance of Lp and its interaction with parameter ME. For example, inspecting Fig.8.10 becomes clear that utilizing $max(ev_1)$ coupled with $p_b^{1,2,5,6,8,9}$ and for $NC > 10$ will most often completely skip the backward pass. Such behavior disregards the collective exchange of information, and converges to the performance most similar to standalone underlying heuristic. On the other hand, when coupled with $p_b^{0,3,7}$, method $max(ev_1)$ does not have the same influence. Contrary, the condition (1) will most likely never be satisfied. In this case Fig. 8.11 provides information on mechanism that generates large number of recruiters for $p_b^{0,3,7}$. With even larger confidence, condition (2) is satisfied for these three loyalty functions. The most revealing are profiles that almost coincide on both figures (for p^0, p^3, p^7), indicating negligible interaction of these loyalty functions and the evaluation process.

Another conclusion concerns method $\max(ev_2)$. When we exclude $p^{0,3,7}$ the method satisfies least likely any of the aforementioned extreme conditions. In Fig. 8.9 we observe that the average number of recruiters per iteration is lower compared to other two

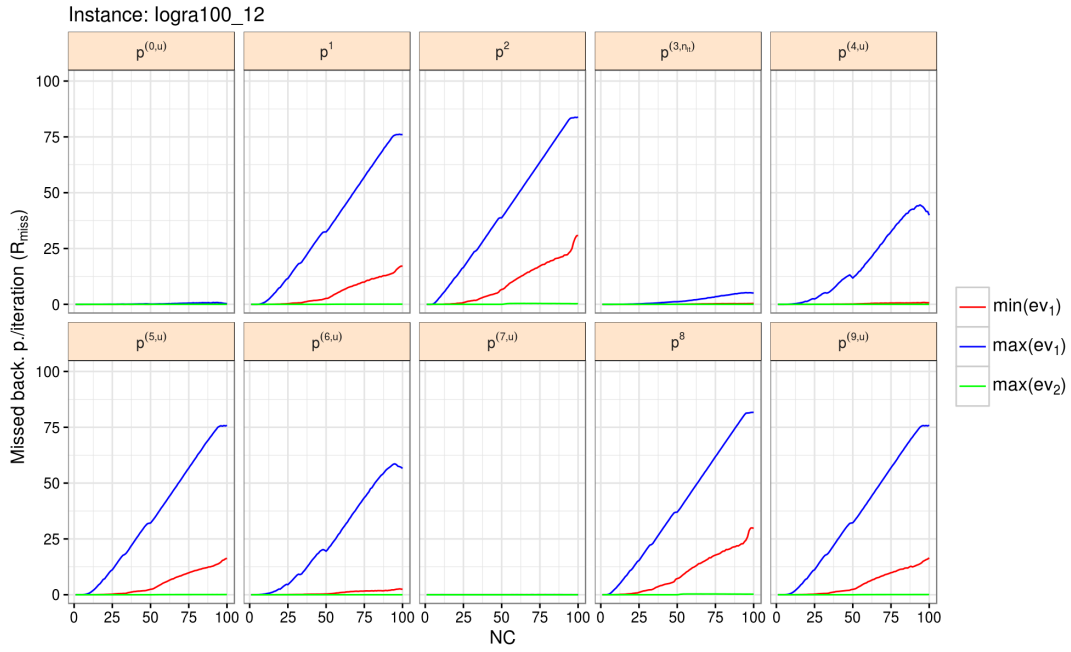


Figure 8.10: Propagation of variable R_{miss} (average number of misses per experiment) reported for $B = 10$, $NC \in [1, 100]$ and test instance *Iogra100_12*.

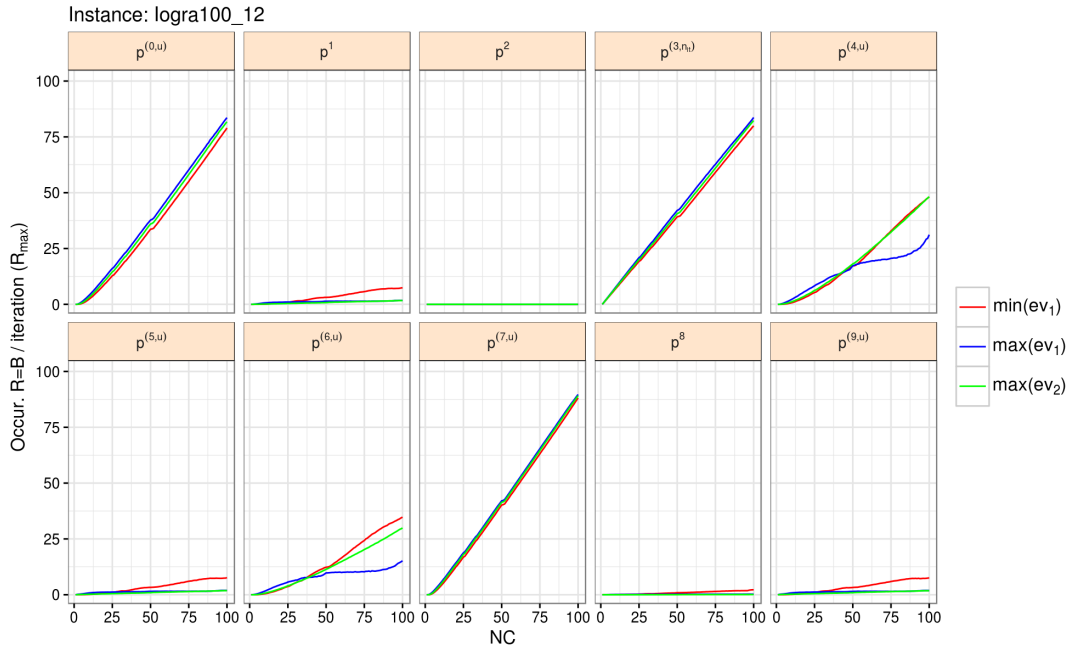


Figure 8.11: Propagation of variable R_{max} (average number of times condition $R = B$ has occurred in backward pass per experiment) reported for $B = 10$, $NC \in [1, 100]$ and test instance *Iogra100_12*.

evaluation methods and that is sensitive to choice of loyalty function. From the same figure, $\min(ev_1)$ represents a sort of in-between case: this method exhibits balance be-

tween number of recruiters and uncommitted bees, and both extreme conditions occur, however, depending on the choice of loyalty function. In Fig. 8.10 functions $p_b^{2,8}$ satisfy the extreme condition (1) more often then the other loyalty functions, whereas in Fig. 8.11 we see that the two will almost never reach the maximal number of recruiters. Functions $p_b^{4,6}$ exhibit very similar behavior in the three reviewed figures. Concretely, from Fig. 8.10 $\min(ev_1)$ does not lead towards misses of the backward pass, whereas from Fig. 8.11 we detect a small increase in the maximal number of recruiters per iteration as NC is growing. In addition, the later trend is more prominent for $p_b^{4,6}$ than for $p^{1,2,5,8,9}$.

We refer to questions posed about $\max(ev_1)$: there are two different mechanisms that made this method robust to the choice of loyalty function, both indicating large number of recruiters per iteration, however, completely separated. For example, in case of p_b^0 method $max(ev_1)$ doesn't induce values that would lead towards *misses* of backward pass. However, the loyalty function is very conservative and therefore causes very often for variable R to reach maximal size of population. The same conclusion holds for p_b^3 and p_b^7.

To conclude, the influence of loyalty functions on the solution quality of BCOc algorithm can be categorized by observing similarities of graphics in Figs. 8.9–8.11. Regardless of method of evaluation, the loyalty functions can be grouped into four categories as illustrated in Fig. 8.12. It is also expected that loyalty functions within each of these categories should exhibit similar tendencies in relation to their performance on other optimization problems.

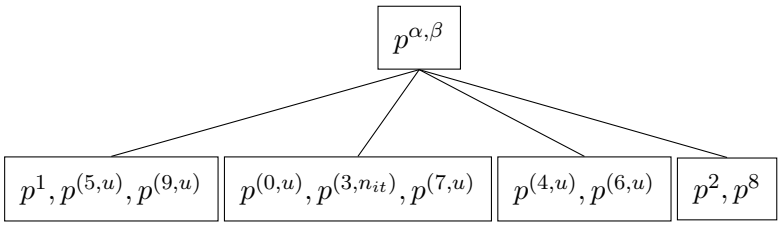


Figure 8.12: Grouping of loyalty functions regarding the similarities exhibited during the recruitment process.

8.9 Final remarks

Beside the parameter ME, as the most prominent parameter of any meta-heuristics, we may state that in case of BCOc the most important and influential parameter is Lp. We base our conclusion on the following: (a) loyalty functions $p^{0,3,7}$ generate similar outcomes regardless of the method of evaluation, indicated by values in basic tables and the shape of surface plots in Figs. 8.2, B.5 and B.6; (b) a correct choice of loyalty function, coupled with parameter tuning, produces the best outcome. The results hold in case of study founded on the average solution quality. To obtain the best possible quality of solutions, the quantitative BCOc parameters B and NC are equally important. With regard to the sensitivity of the response, the two parameters also demonstrated equal importance. The conclusions is based on existence of interactions between them, as shown in Figs. 8.2, B.5 and B.6. In particular, in Figs. B.5 and B.6

we detect deterioration of average solution quality along values of B or NC or both. In Fig. 8.2 $\min(ev_1)$ we may also observe high non-linearity of response landscapes.

There are several reasons to conduct experimental analysis for the two stopping criteria: N_{it} and T. Appointing these categories of the stopping criterion to the experimental tests has primarily effected an influence of the quantitative parameters to the BCOc performance. For example, an increase in the values of B degenerates the quality of solutions at certain levels of Lp. The conclusion is easier to deduce when the maximal allowed CPU time is imposed. Another advantage of our approach is the assessment of the reliability of the parameter Lp at different levels of the parameter SC and at ME. For $\min(ev_1)$ the relation between the performances of the BCOc instances at different levels of SC does not change, i.e., a loyalty function that performed well under N_{it} also demonstrates a good performance under T. In particular in Tables 8.13, 8.14, 8.17 and 8.18 it is shown that for both stopping criteria the method of evaluation $\min(ev_1)$ influences loyalty functions p^2 and p^8 to report the best results for large B and large NC. However, the conclusion changes for $max(ev_{1|2})$. Here, p^2 and p^8 reported best responses for different values of NC and B w.r.t. SC. Moreover, for $max(ev_{1|2})$ the best performance is recorded for class II loyalty functions. Therefore, an important conclusion of the studies is the relation between all components of the BCOc algorithm, especially Lp and the parameters B and NC. Considering the question of robustness, the BCOc algorithm is quite sensitive to the properties of problem instances, however, certain rules are possible to establish (e.g. for p^2 and p^8). Therefore, the robustness of BCOc is possible to achieve after careful structural tuning.

We address the research question Q_5: if setting stopping criterion as $N_{it} = 100$ do we obtain enough information about the performances for allowed maximal CPU. The result of our study show that the best configuration $(\min(ev_1), p^8, B \geq 15, NC \geq 90)$ found in N_{it} study also applies in T study. There are other cases for which the best configuration found under N_{it} was generated under T. For Iogra100_16 function p^5 found the best configuration pair $(B, NC) = (20, 97)$ under both types of stopping criteria. However, it is expected that providing more iterations would yield better result of other configuration pairs.

The notion of practical difference can be set arbitrary as it depends on the observer. As the measure is mean value of the set of solutions, we have concluded that the half of a unit time provides good practical difference (pg. 125). To capture such differences, an effect size of RANOVA test may be used.

8.10 Chapter summary

This chapter is devoted to comprehensive experimental study of BCOc, implemented for $P||C_{max}$. It presented many new results concerning the general behavior of the BCOc algorithm. List of conclusions emphasize the importance of careful structural and parameter tuning. In the chapter, two statistical tests were utilized for within-subject comparison of data in order to test the hypothesis of equivalence of means between loyalty functions. Selection of best performing loyalty function within each group is conducted using post-hoc test in R via two-sided Dunnett's test or pairwise test. A number of conclusions have been reached based on the results of multiple comparison, i.e., post-hoc tests. More precisely:

- We investigate several known heuristic methods dealing with the corresponding problem.
- We provide information about the structure of problem instances with box-plots of processing times distributions.
- We conduct experimental tests in order to compare several heuristic algorithms and determine the best candidate to be used within the implementation of the BCOc algorithm.
- Results of statistical and visual analysis are given for two independent studies w.r.t. the type of stopping criterion.
- We introduce different case studies regarding qualitative BCOc parameters. We are interested in relations between parameters Lp and ME.
- We presented a series of statistical tests, where we examine effects of qualitative BCOc parameters. The tests are accompanied with a couple of graphics.
- Our conclusion is that regarding parameter Lp for the $P||C_{max}$ problem we recommend Class I loyalty functions.

CHAPTER 9

Development and experimental analysis of BCOi

This chapter is devoted to design and experimental analysis of BCOi for 3-SAT. We explore different stages of the development of the BCOi algorithm through analysis of experimental results generated by different BCOi's parameter configurations. The importance of selecting suitable modification rules in the forward pass of BCOi is addressed. We propose a simple algorithm design for 3-SAT with familiar procedures of transformations of complete solutions. In particular, three elementary 3-SAT solvers are explored: WalkSAT and two random walk techniques. The advantage of the BCO model over a standalone solver is its exploitation of population of solutions and the utilization of reasoning when comparing different solutions. Moreover, the population serves to maintain knowledge that guides the search towards promising areas of the search space. Results show that carefully designed method of evaluation can improve an overall success of the BCOi algorithm.

At the beginning of the chapter we describe the properties of the problem instances with *clause-to-variable* parameter. We then compare three 3-SAT solvers to determine the order of magnitude a sophisticated 3-SAT solver, such as WalkSAT, outperforms the random walk techniques. The second half of the chapter provides description of two BCOi algorithm implementations: BCOi with random walk (`randBCOi`) and BCOi with walksat modifications (`WalkBCOi`). Each algorithm is analyzed in a separate section and the main conclusions are drawn from graphics of response values. Unlike the previous chapter, we have not addressed questions related to the dynamics of recruitment. Majority of the results are for the first time presented in this thesis.

9.1 Sensitivity analysis of the BCOi algorithm

Our main research question is if BCOi framework can improve performance of the corresponding underlying solvers. We have tackled this problem the first time in [JK16a], and, here, we extend the analysis by exhausting the list of the BCOi parameters. In addition, we utilize different performance measure.

Sensitivity analysis of BCOi is not as comprehensive as for BCOc. Firstly, we do not employ statistical analysis. Secondly, we investigate the algorithm's behavior on a problem-set, as opposed to investigations conducted for each problem instance seperately. Contrary to the common approach to 3-SAT, we search for an assignment (model) that satisfies a given 3-CNF formula for which we *a priori* know that is satisfiable. Accordingly, we deal with a problem that coincides to some extent with the well-known

MAX-3-SAT (Section 2.3.3). Problem instances that are originally designed for 3-SAT are utilized. In particular, we exploit the well-known SATLIB library of 3-SAT instances [Hoo00b, Stü01]. Graphics of the results are based on the two most common response values: average number of flips and average number of unsatisfied clauses. Conclusions of the visual analysis shows that the BCOi algorithm is influenced by different algorithm components, especially by modification rules in the forward pass. Before demonstration of the main conclusions, we test two focused random walk and walksat algorithms (see Section 2.3.4). Another objective is to identify parameter properties, such as type and value of the stopping criterion, required by instructions in Section 7.3.5. Here, we also distinguish two cases regarding the type of stopping criterion. Comparison between three 3-SAT solvers was conducted under a maximal allowed CPU time (cutoff time, T). The cutoff time serves as a constraint to avoid the consequences of ceiling and floor affects in the study of BCOi. However, results of investigation of the BCOi instances were generated by setting the maximal number of flips (MAXFLIPS), similar to maximal number of evaluations discussed in previous chapters. Statistical tests are not employed since response values are numerically far, thus, easy to distinguish in practical terms.

9.1.1 Research goals

Here, we briefly list research questions and goals of the dissertation. Our focus is primarily to address the question if the BCOi framework contributes to the improvements over standalone heuristics. A choice of the best parameter configuration represents another line of the research. In particular, we postulate that the optimal performance of BCOi depends on the correct selection of two qualitative BCO parameters: loyalty and evaluation function. In contrast to study in Chapter 8, we do not conduct a scaling analysis [Hoo09], i.e, emphasize influence of problem instances. The purpose of this research is summarized as follows:

- Q_1 Identify the most influential BCOi paramater.
- Q_2 Compare two evaluation functions ev_1 and ev_2 for `WalkBCOi`.
- Q_3 Identify the most successful loyalty function(s) and the reasons behind its(their) success.
- Q_4 Compare conclusions among `randBCOi` and `WalkBCOi`.
- Q_5 Investigate behavior of the BCOi algorithm w.r.t. underlying heuristic rules.

We believe that the answers to these research questions might help in design of the BCOi algorithm for other similar combinatorial problems.

9.1.2 Swarm intelligence for SAT

Meta-heuristics are in the literature recognized as incomplete SLS solvers. As a result, they are suitable for model-finding problems because the stochastic algorithms can not be utilized alone to prove unsatisfiability. Population-based meta-heuristic algorithms have already been employed to deal with MAX-SAT [Fra94, Ran98, Abb01, Stü01, Vil07]. In some of the articles it has been shown that the meta-heuristic algorithm does not represent the best choice for MAX-SAT. For example, analysis of a

simple genetic algorithm (SGA) for MAX-3-SAT problems reveals that SGA does not exhibit good results [Fra94, Ran98]. [Fra94] implemented parallel SGA and reported that parallelism does not enhance the performance of SGA. The conclusions motivated further investigation to find a potential reason. [Ran98] tries to identify a property of a problem instance that leads away from the global optimum, known as *deceptiveness* of MAX-SAT problems. The authors conducted several experimental tests on the problem set containing three randomly generated 100-variable MAX-3-SAT problems of different hardness ($\alpha = \{3, 4.3, 6\}$). The conclusion is that, beside being deceptive, the problem instances contain equally good regions which direct SGA to converge towards different regions of the search space. Poor performance of GA for MAX-SAT problems has been reported throughout the literature which inspired [Got02] to employ additional techniques in EA. Authors of [Got02] proposed ASAP and the utilization of local search and adaptive mechanisms. They emphasize the importance of an appropriate evaluation function and employ *refining functions* that help to distinguish between solutions of the same quality. Other reports of poor performance was reported for ACO. In [Vil07] the authors implemented a novel ACO algorithm, utilizing an adaptive fitness function (commonly used in GA) to escape local optima and show that the efficiency of population-based meta-heuristics might be improved [Vil07, pg. 352]. [Mor12] demonstrated that ACO with adaptive fitness function outperforms the GA implementation. Beside the TS algorithm for MAX-SAT, a single-solution meta-heuristics have shown to be efficient for probabilistic satisfiability problems (PSAT). For example, the VNS performed well on PSAT compared against the GA [Jov05].

Regrading the bee-inspired algorithms in [Abb01] authors show that MBO performs better than WalkSAT. A committee of heuristics is incorporated in the MBO algorithm: WalkSAT, random walk, randomflip, random new and 1-point crossover and compared against the standalone implementation of WalkSAT. A similar investigation of an improved MBO (employing variation of the annealing function) was tested against different 3-SAT solvers, such as GSAT and random walk, and results presented in [Teo01]. The authors concluded that MBCO outperforms other standalone heuristics. However, it can be easily demonstrated that different implementations of WalkSAT (as well as other solvers) exhibit different performances, being sensitive to the type of data structures and other implementation tricks. This knowledge has motivated us to discover a code for WalkSAT developed by Henry Kautz[(1)]. The implementation may be executed either as a standalone algorithm or within other state-of-the-art implementations.

9.1.3 Problem instances

As previously mentioned, experimental analysis of the BCOi algorithm is conducted on uniform random 3-SAT family of problem instances which consist of randomly generated 3-CNF formulas [Hoo00a, Hoo00b]. The instances belong to the SATLIB library[(2)] and represent a well-established problem-set of *unweighted* uniform k-SAT problem instances to test new algorithms. SATLIB library was constructed to deal with the decision variant of k-SAT, however, according to [Stü01] is utilized for MAX-k-SAT. We refer to Section A.4.1 (pg. 265) for information about the motivation behind the creation of SATLIB and to Section A.4.1.1 for classifications of k-SAT problem instances w.r.t. their

[(1)] http://www.cs.rochester.edu/u/kautz/walksat/
[(2)] http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

hardness. Hardness of a particular problem instance can be measured as a function of the expected number of steps needed to find a solution (an assignment is found so that formula F is true) [Hoo98b, pg. 94].

9.1.3.1 Clause-to-variable ratio

Analysis of clause-to-variable ratio of random k-SAT problems (α) was important for discovery of the *phase transition* phenomenon [Coo97]. The phenomenon reveals the existence of a threshold for α, *critical value* α_c, so that really difficult instances from an algorithmic perspective are those close to α_c [Mer06]. The most important aspect of this phenomenon is the transition from formulas that are satisfiable to unsatisfiable formulas. According to [Méz02a], regardless of the number of clauses and variables, random k-SAT problems are generally satisfiable for small α and generally unsatisfiable for large α. The value of α_c has been reported throughout the literature [Méz02a, Méz02b, Mer06]. The latest reported value for $k = 3$ is $\alpha_c = 4.267$ [Mer06]. The additional information about the α and its critical values might be found in [Hoo98b, Hoo98b, Brg03, Hoo05, Gen10a].

Creators of SATLIB focus on maximizing the hardness of solvability of problems, thus, propose instances with the corresponding phase transition region [Hoo00b, pg. 4]. It seems they have succeeded since, according to [Mit92, pg. 3], the transition region for $n = 20$ (small) occurs at $\alpha_c = 4.55$, for $n = 50$ $\alpha_c = 4.31$ and for $n = 140$ the transition happens when $\alpha_c = 4.3$.

9.1.3.2 Procedure for generation of hard 3-SAT instances

The procedure to randomly generate 3-CNF formulas is elaborated in [Hoo00b] and follows steps from [Mit92]: m times (the number of clauses) 3 variables are selected (from the set of n available) and then each is negated with the probability of $1/2$. The procedure assures that clauses, containing multiple copies of the same literal (or multiple copies of a variable and its negation), are discarded. Values of the parameters n and m induce different distribution of random 3-SAT instance, thus, different values of clause-to-variable ratio. We consider only satisfiable problem instances that have been successfully solved by various SLS algorithms. Based on the critical value, in Table 9.1 we review ten problem sets from SATLIB. The problem instances are arranged w.r.t. n and m (columns 3 and 4). As shown in Table 9.1, α decreases as n increases and does not drastically change for different n, m. Secondly, it is easy to distinguish between the set of easy and hard problem instances. Among the reported instances, class uf20-91 contains 3-CNF formulas with 20 variables and 91 clauses ($\alpha = 4.55$), is considered the easiest and originally contains 1000 different test instances. Class uf250-1065 contains 100 randomly generated instances with 250 variables and 1065 instances. Due to time constraints, we chose the top three problem-sets from Table 9.1 to conduct a comparison study between the candidate 3-SAT solvers.

9.1.3.3 Properties of SATLIB 3-SAT instances

A high variability in response values (i.e, the hardness of the corresponding problems) of 6 or more orders of magnitude of various SLS algorithms dealing with randomly generated 3-SAT instances with the same critical value, has been reported in the literature

Table 9.1: SATLIB 3-SAT satisfiable instaces.

class	setsize	n	m	α
uf20-91*	1000	20	91	4.55
uf50-218*	1000	50	218	4.36
uf75-325*	100	75	325	4.33
uf100-430**	1000	100	430	4.30
uf125-538	100	125	538	4.31
uf150-645	100	150	645	4.30
uf175-753	100	175	753	4.30
uf200-860	100	200	860	4.30
uf225-960	100	225	960	4.27
uf250-1065	100	250	1065	4.26

* Test-instances in the experimental study of standalone solversf.
** Test-instances in the experimental study of BCOi.

[Gen10a, pg. 617]. The variability caused by SATLIB 3-SAT instances is elaborated in [Hoo98b, Hoo05] [Hoo98a, pg. 108]. [Hoo05, pg. 214] shows that the inter-instance variability of hardness of uniformly generated 3-SAT instances (i.e., inside the same phase transition region) can be categorized. Namely, three sub-classes are distinguished as *easy*, *medium* and *hard*. [Hoo05] explain that the classification is a consequence of the landscape of the search space. How hard it is to solve a given instance can be further identified with the distribution of solutions. A logical explanation may also be found in the intrinsic property of the solver. To overcome the problems and increase the experimental reliability, performance analysis can be conducted for special problem instances, e.g., representatives of an equivalence class within a benchmark problem-set [Brg03].

We note that the question of SAT problems hardness remains an open topic due to heterogeneous and richly structured distribution of problem instances. Recently, unsatisfied with complexity-theoretic analysis [LB14] have presented *empirical hardness models* (EHM). The model estimates a running time of an algorithm and describes simple relationships between instance properties and the algorithm runtime. The model is not considered in this thesis and is beyond the scope of this dissertation.

9.1.4 Experimental methodology

To ensure fair comparison among investigated algorithms and reproducibility of the experiments we follow rules specified in Section 7.3.5 (pg. 137). For the most part we repeat experimental steps from Chapter 8: blocking of $seed$ and number of repetitions n_{run} is appointed to large values to secure stable measure of the performance ($seed$ coincides with experiment indices). The first case study deals with an empirical analysis of the candidate solvers. The runs in the experiments are repeated 1000 times due to large variability in the response values and to establish the number of repetitions for study of BCOi. The result showed that the n_{run} for evaluation of the BCOi algorithm

can be set to 100 as it avoids over-usage of computational resources.

Because the performance of the considered 3-SAT solvers is well established in the literature, extensive analysis of random walk and walksat heuristics was not conducted as was done for $P||C_{max}$ in Chapter 8. Namely, the objective is to to emphasize the magnitude of success of walksat algorithm against the random walk technique. Another objective of our experiment is to test theoretical results regrading number of restarts (reinitialization) of the random walk ($3n$) proposed by [Sch99]. The empirical analysis is conducted under the same computer environment as for BCOc (see Section 7.3.5.2, pg. 137).

9.1.4.1 Performance measure

Total number of flips (n_{flip}) necessary for a solver to find a model of a given problem instance or reach a stopping criterion, together with the corresponding descriptive statistics over multiple runs, represents a typical measure in the literature when dealing with MAX-3-SAT problems. As mentioned previously, n_{flip} exhibits high variability in case of uniform random problem instances [Hoo98a]. Therefore, to evaluate the performance of our solvers we opted for average value of n_{flip} computed as $\overline{n_{flip}} = \sum_{s=1}^{n_{run}} n_{flip}$. The response value assures an objective comparison between reported results from different computer architectures. If for a value of $seed$ solver did not find a satisfiable model, we employ a number of unsatisfied clauses variable n_{uns} and, consequently, average value of unsatisfied clauses $\overline{n_{uns}} = 1/n_{run} \sum_{s=1}^{n_{run}} n_{uns.}$. In [JK16a] maximal number of flips M_{flip} is applied as a measure of performance, computed as $M_{flip} = \max_{seed \in [1,n_{run}]} n_{flip}$.

9.1.4.2 Stopping criterion

Two types of stopping criteria are considered: maximal allowed CPU time (T) and maximal number of flips (MAXFLIPS) . From the literature we were not able to retrieve running times of a simple random walk solver for instances in Table 9.1. Therefore, we restricted maximal allowed CPU time arbitrary and set it to $5s$. In the case of the walksat solver our results showed that it finds satisfactory assignments for the majority of the considered problem instances in a very short time (e.g. less then 0.01 seconds for $n \leq 75$). The random walk requires large amount of a running time until it finds a model. However, increasing T is not required as it will be shown in the Section 9.2. The cutoff time T was not imposed in the case study regarding the BCOi algorithm. Namely, utilized time procedure `gettimeofday` is not able to distinguish work of computer system background processes, thus, we focused on minimization of systematic errors during the evaluation. We employ MAXFLIPS, appointed to 10^6 and elaborate in greater detail in Section 9.2.

9.1.4.3 Size and choice of the problem-set

The number of problem instances for BCOi is larger than for the consider scheduling problem of previous chapter. Moreover, comparison between the solvers is conducted on the entire problem-set. The problem-sets can be distinguished by the number of variables (n) and number of clauses (k), as presented in Table 9.1. There are several reasons to expend the problem-set. Firstly, uniform random 3-SAT SATLIB instances are commonly employed in the literature and there exist a number of research articles and

dissertation thesis offering different perspectives about efficient solvers. The reason we test BCOi on the entire problem-set is to ignore particularities of a problem structure. In addition, walksat solver exhibits robustness while dealing with the SATLIB problem instances and the same is therefore expected for `WalkBCOi`.

In this thesis we distinguish three problem instances of the same size as representatives of easy, medium and hard ones, in the same manner as reported in [Hoo98a]. Selection of the instances is, to some extent, founded on results of the walksat algorithm (`wsat`) reported in [Hoo98a, pg. 93], summarized in Table 9.2.

Table 9.2: Results of `wsat` for uf100-430 [Hoo98a].

hardness	mean ($\overline{n_{flip}}$)	stddev	median	Q_{75}	Q_{90}
easy	177.86	135.78	139	206	336
medium	1,877.78	1,776.33	1,333	2,510	4,133
hard	86,773.44	90,538.08	56,666	120,583	198,109

Tables of cumulative distributions are employed in the first case study to graphically demonstrate difference between the candidate heuristics. The graphics present cumulative distributions of N_{flip} over the problem-set. In case of BCOi we utilize a set of instances with 100 variables and 430 clauses, provided in SATLIB library as *uf100-430*. The set originally consists of 1000 instances, therefore, to decrease the total running time of the experiment and contribute do the reproducibility, we have opted for the first 100 instances (uf100-01.cnf – uf100-0100.cnf)

9.1.4.4 Solution representation

We briefly describe structure of the solution, and the corresponding auxiliary variables utilized in the implementations of all candidate solvers and BCOi instances. The solution is represented as a vector $A = (a_1, \ldots, a_n)$, where the element a_i contains the value of the i-th variable. Considering the unsatisfied clauses, two vectors are applied: $F = (f_1, \ldots, f_m)$, a binary vector that indicates if clauses are (not) satisfied, and dynamic vector $W = (w_1, \ldots, w_m)$ containing indices of all the clauses that are unsatisfied. Considering the $break$ condition of the WalkSAT, after flipping i-th variable, the number of clauses that become unsatisfied is saved in the vector $C = (c_1, \ldots, c_n)$.

At the end of an iteration data structure (A, C) contains the obtained solution and the information about the unsatisfied clauses (if satisfying assignment is not obtained).

9.2 Candidate heuristics for 3-SAT

In this section we compare three 3-SAT solvers. The focused random walk belongs to the most fundamental approaches dealing with satisfiability problems. Therefore, we consider pure random walk elaborated in Sec. 2.3.4.1. Another version of a random walk technique is the Schöning's algorithm (Fig. 2.3, pg. 33), selected in order to test results in [Pre04, Bal14a] stating that Schöning's procedure does not perform well in practice, despite of theoretical proof of its convergence [Sch99]. We denote our implementations of the two random walk techniques for 3-SAT as: `Rand` and `SchRand`.

In the dissertation we utilize a code of the walksat algorithm developed by Selman and Kautz [Sel95]. The choice is supported by the fact that the code is being regularly maintained in order to increase the performance by involving better parameter structures and some other implementation tricks. Therefore, it contains good properties regarding a performance and robustness for various satisfiability problems. The code is also used as the basis for development of more sofisticated SAT solvers [Bal14a]. The latest version (51) is provided freely online [(3)]. In addition, the authors report that the version 51 incorporates optimization fixes suggested by Donald Knuth[(4)] making it 20% faster than the previous version. Originally, the code by Selman and Kautz contains implementations of 7 different k-SAT solvers: *random*, *walksat*, *tabu*, *novelty*, *rnovelty*, *novelty+* and *rnovelty+*. Heuristics such as *random*, *walksat* and *novelty* are considered as simple solvers, commonly employed as a building stone for other heuristic algorithms. As the standalone heuristics they perform well and *novelty* is considered among the best for 3-SAT SATLIB instances since it utilizes more knowledge than the other two. However, *novelty* was not considered in our study because it requires more memory then the other two and, therefore, we believe it is not suitable for BCOi.

9.2.1 Experimental evaluation of candidate solvers

The main reason behind comparison of `Rand`, `SchRand` and `WalkSAT` is to illustrate the size of the extent by which walksat outperforms two random walk procedures. Following recommendations in Section 7.3.5 (pg. 137) an experiment consists of 1000 independent runs of a particular solver and *seed* is controlled. We distinguish two case studies: on a particular instances and on complete problem sets. In the first study three types of instances are chosen, easy, medium and hard from *uf50-218*. An easy instance requires the least amount of the computational resources, measured by n_{flip}. A hard instance represents the hardest within the experimental setup. Medium instance is the hardest to select and represents an in-between case. The results of single comparisons are given in Tables 9.3. Under the provided stopping criteria all three solvers manage to find solutions under given conditions, thus, we provide descriptive statistics for response value n_{flip} in Table 9.3. We conclude on several observations. The span of n_{flip} values, required for random walk techniques to solve easy and hard instance, is large. This illustrates the inefficiency to deal with small problem instances and already provides information about the performance for larger problem-size instances. In addition, `WalkSAT` required small amount of computational resources, thus, represents the best performing solver on the class *uf50-218*.

Experiments in the second study were conducted for three problem-sets of different size: *uf20-91*, *uf50-218* and *uf75-325*. The problem sets are described in the literature as small- to medium-sized since satisfiable assignments are fast reachable by majority of 3-SAT solvers on modern computer systems. Performance measure in this case study is $\overline{n_{flip}}$. To highlight the range of the results during an experiment and for various problem sets, we utilize RLDs for $\overline{n_{flip}}$ (see Section 7.2.2) and we compare them in Fig.9.1. According to long tails in Fig. 9.1 the class uf75-325 of problem instances exhibits high variability in case of random walk solvers. The `WalkSAT` exhibits lower

[(3)] `https://www.cs.rochester.edu/u/kautz/walksat/` Version 51

[(4)] Information about the improvement of last version is taken from the comment section of Walksat source code.

Table 9.3: Descriptive statistics for n_{flip} reported by three candidate solvers for three problem instances in class uf50-218, $T = 5[s]$.

Solver	hardness	mean	stddev	median	Q_{75}	Q_{90}
`Rand`	easy (uf50-0635.cnf)	368.18	309.55	276	481.00	740.20
	medium (uf50-0105.cnf)	16,061.19	15,125.39	11,225	22,693.00	37,733.10
	hard (uf50-0690.cnf)	884,505.20	951,193.63	574,474	1,231,330.50	2,046,741.50
`SchRand`	easy (uf50-0635.cnf)	617.84	591.88	422	862.25	1336.20
	medium (uf50-0105.cnf)	38702.16	38672.04	27,776	53603.25	88766.70
	hard (uf50-0745.cnf)	1,597,364.01	1,611,398.93	1,117,166	2,228,060.25	3,613,145.70
`WalkSAT`	easy (uf50-0543.cnf)	64.17	43.87	52	83.00	120.00
	medium (uf50-0105.cnf)	596.03	640.47	399	770.50	1376.80
	hard (uf50-0197.cnf)	4995.69	5617.69	3256	6978.00	12025.00

variability, however, according to [Hoo98a, pg. 86], the solver demonstrates large variability for harder problem instances. A summary of descriptive statistics for n_{flip}, is presented in Table 9.4. The table is arranged in the following way. We group results by rows for each instance-set. The first column corresponds to the size of the 3-SAT problem-set. The second column refers to the algorithm, and the third presents average number of unsatisfied clauses for all instances within the test-set (N_{uns}). The next five columns contain data regarding the descriptive statistics of the number of flips needed to either reach optimal solution or satisfy the stopping criterion. Variables Q_{75} and Q_{90} denote the corresponding percentiles of the data. Column $\bar{t}$ shows the average CPU time needed to either solve all problem instances or to reach the predefined stopping criterion, whereas the last column holds standard deviation for $\bar{t}$.

The results in Table 9.4 show that WalkSAT is able to solve all three test-sets of problem instances in a very short time ($\leq 0.01s$). Among random techniques, it can be concluded that `Rand` outperforms `SchRand` according to the number of flips (mean, median, Q_{75}, Q_{90}) required to either reach a solution (*uf20-91* and *uf50-218*), or satisfy stopping criterion (*uf75-325*). Therefore, we confirm results from the literature [Pre04,

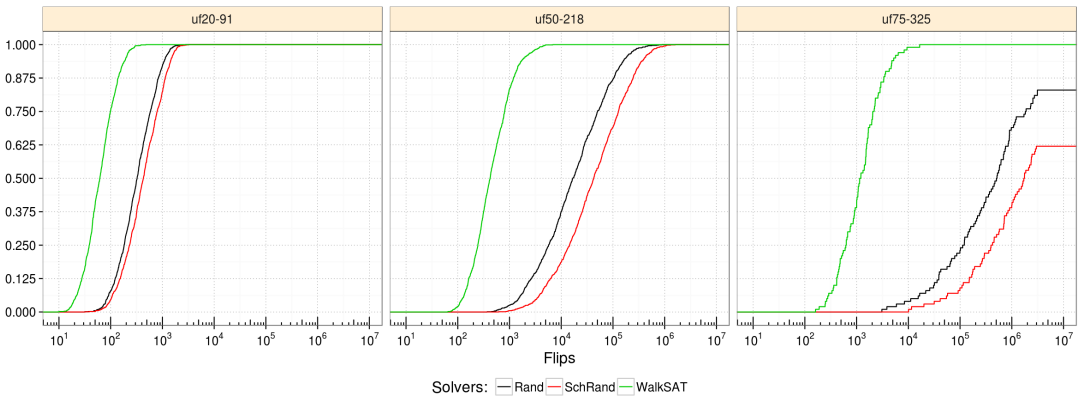


Figure 9.1: Semi-log plots of cumulative distribution of $\overline{n_{flip}}$ for three 3-SAT solvers and three problem sets. The x-axis shows $\overline{n_{flip}}$ based on 1000 runs/instance and stopping criterion $T = 5s$.

Table 9.4: Results for `Rand`, `SchRand` and `WalkSAT` for SATLIB problem-sets. Optimality criterion is an average number of flips.

Problem			descriptive statistics for N_{flips}						
class	solver	N_{uns}	mean	stddev	median	Q_{75}	Q_{90}	$\bar{t}$[s]	s.d.
uf20-91	`Rand`	0.00	434.52	353.84	324.01	585.76	919.88	0.00	0.00
	`SchRand`	0.00	574.37	460.62	428.30	790.83	1229.77	0.00	0.00
	`WalkSAT`	0.00	78.00	57.45	61.73	98.66	154.82	0.00	0.00
uf50-218	`Rand`	0.00	44864.50	76822.37	16765.03	50222.59	116259.68	0.01	0.02
	`SchRand`	0.00	104496.10	159881.57	42396.87	127105.71	275474.68	0.03	0.04
	`WalkSAT`	0.00	653.32	711.56	421.93	806.02	1347.98	0.00	0.00
uf75-325	`Rand`	0.22	1678433.80	2887571.68	511304.80	1844154.11	4578383.66	0.35	0.61
	`SchRand`	1.74	3842645.64	4760262.65	1731620.82	5911339.67	11311854.43	0.91	1.13
	`WalkSAT`	0.00	1850.35	2222.88	1177.91	2086.47	3700.32	0.00	0.00

Bal14a]. Although being theoretically verified and analyzed, `SchRand` seems not to be of practical use. Coupled with Table 9.4, the results presented in Fig.9.1 show that `WalkSAT` exhibits a clear dominance over random walk procedures by a couple of orders of magnitude.

9.2.2 Final remarks and conclusions for candidate heuristics

Comparison and evaluation of three 3-SAT solvers was conducted within two case studies. The first case study follows the line of research from the previous chapter and considers individual problem instances: easy, medium and hard. It turns out that the medium-size problem instance is the representative of the majority of problem instances. As a preparation for experimental evaluation of the BCOi configurations, the second study was conducted on the complete problem sets. Results of the both studies have confirmed two observations reported in the literature about the random problem instances. Namely, the existence of long tails in cumulative distributions (Fig. 9.1) demonstrates variability in hardness of problem instances of the same class (same number of variables). Furthermore, we show that shows that the `WalkSAT` solver outperforms the `Rand` algorithm.

Results of the conducted experiments demonstrate that appointing MAXFLIPS$=10^6$ is sufficient for empirical analysis of the BCOi algorithm for harder class, uf100-430. The `WalkSAT` solver is able to find model for different values of $seed$, therefore, we expected that `WalkBCOi` exhibits equivalent or better performance. Contrary, we have detected an inefficiency of random walk for problem instances of class *uf50-218*, thus, the stopping criterion would not not be enough for `randBCOi` to solve uf100-430 problems. Therefore, we chose to conduct the experimental evaluation of the `randBCOi` on two classes of problem instances *uf50-218* and *uf100-430*. Furthermore, we analyze if the conclusions about the BCOi parameter configurations will change w.r.t. the problem-size. The selection of the problem sets was also inspired by reports in [Hoo98a].

9.3 Development of BCOi for 3-SAT

In this section the objective is to analyze conditions under which working with the population of solutions demonstrates an advantage over a standalone heuristic. A simple approach would be to start a solver from different positions of the solution space and restart the execution after a predefined restarting criterion is satisfied. In this case the search would be highly dependent on the quality of initialization points. To make it robust, we focus on design of the BCOi algorithm for the 3-SAT problem that benefits from certain reasoning when comparing different solutions. Furthermore, we employ the main steps of two considered heuristic rules, examined in previous section, and test two version of the BCOi algorithm: `randBCOi` nad `WalkBCOi`. Success is measured by the average number of flips until reaching satisfactory assignment or stopping criterion. Occasionally, average number of (un)satisfied clauses is utilized if the solver required a maximal number of flips to satisfy the stopping criterion and because of large number of problem instances and the variability that BCOi algorithm exhibits across the problem set.

9.3.1 Design of the BCOi algorithm

Contrary to other implementations, the BCOi algorithm for 3-SAT disregards initialization of solutions at the beginning of each iteration. Instead, an initial assignment is appointed to each bee once before the start of the search. Consequently, all solutions reported at the end of an iteration become the initial solutions for the next iteration. Psuedo-code of BCOi for 3-SAT is presented in Fig. 9.2. Total number of transformations within a forward pass of BCOi is not restricted for 3-SAT as it is, e.g., for p-center problem [Dav11a]. According to [Dav11a] the restrictions originate from the number of centers. However, restrictions for 3-SAT are not as obvious. In particular, we control transformations during the forward pass of BCOi, with new parameter NCT. An iteration of BCOi completes if a solution is found or if each bee performs NCT solution transformations.

Without the typical initialization phase at the beginning of an iteration, the number of successive forward/backward moves during one iteration, and controlled by parameter NC, does not directly influence the search trajectory. Namely, class I loyalty functions do not rely on values of NC. However, we utilize NC to control the upper limit of the counter u, being important for Class II loyalty functions. As a result, parameter NC does not inflict the total number of evaluations.

Parameter space of BCOi is summarized in Table. 9.5, where n denotes the number of Boolean variables. All the values of the BCOi quantitative parameters are integers. Because of the expected overhead during the exchange of information in a recruitment phase of BCOi, we investigate small number of the artificial bees. Values of quantitative BCO parameters NC and NCT have been adjusted to accommodate a requirement in which the running time should not be longer than a week. However, our objective is to cover diversified regions of the parameter space. Therefore, NC takes discrete steps and its upper limit has been determined with the help of analytical expression of loyalty functions in Section 4.2.3.1. In Figs. 4.10–4.17, after NC reaches 60 the loyalty probability p_b^u approaches 1, resulting with behavior that might produce similar results. Namely, it becomes almost certain that a bee remains loyal to its generated

Initialization: Read input data, formula F, number of variables x, number of clasues.

Provide random assignments to each each bee.
Do
// **forward pass**
(1) *For* $(b = 0; b < B; b++)$
(a) *For* $(i = 0; i < NCT; i++)$
(a.i) Flip the variable using a heuristic.
(a.ii) *if* $(F(x) = TRUE)$ stop.
// **backward pass**
(2) *For* $(b = 0; b < B; b++)$
Evaluate the solution of bee b;
(3) *For* $(b = 0; b < B; b++)$
Loyalty decision for bee b;
(4) *For* $(b = 0; b < B; b++)$
If (b not loyal), choose a recruiter by roulette wheel.
Update x_{best} and $f(x_{best})$
While stopping criterion is not satisfied or solution is found.
return $(x_{best}, f(x_{best}))$

Figure 9.2: Pseudo-code for BCOi for satisfiability problem.

solution. According to results in Section 8.8, to avoid disregarding the main principles of recruitment and to support exploitation, the upper limit on NC domain is set to 60. Because the maximal value may be exceedingly large, we limited the upper value of NCT with the n of the corresponding 3-SAT instance. For example, for 3-SAT problem with 100 variables the BCOi solver performs 100 transformations within one iteration. We wanted to avoid situations in which a solution is found in the first forward pass of BCOi and before backward pass has even begun.

Table 9.5: Parameter space for experimental analysis of BCOi on 3-SAT.

Parameter	**Domain**
evaluation	$numfalse$, $breakcount$
loyalty function	p^i, $i \in \{0, .., 9\}$,
B	$[1, 5]$
NC	$10 \cdot r, r \in [1, 6]$
NCT	$[1, n]$

9.3.2 Backward pass of `randBCOi` and `WalkBCOi`

According to pseudo-code 9.2, numerous implementations of the BCOi algorithm for 3-SAT might be considered: it is sufficient to replace step (a.i) with any transformation of our choice. In the dissertation we chose a simple design of BCOi that implements either focused random walks (`randBCOi`) or walksat (`WalkBCOi`).

The `randBCOi` algorithm incorporates evaluation function that exploits only the number of unsatisfied clauses. We define the evaluation function as $ev_1 = numfalse(b)$, where variable $numfalse(b)$ denotes the number of unsatisfied clauses for each bee b.

In case of WalkBCOi, beside ev_1, we investigate peformance of new evaluation function $ev_2 = breakcount(b)$. The function ev_2 is inspired by walksat and exploits the first decision step of its procedure, i.e., utilizes variable $break$ to obtain a different perspective to quality measure. The goal is to support the efforts of walksat heuristic, contrary to qualifying the goodness of the search space solely on a current number of unsatisfied clauses. The evaluation function ev_2 is described in Figure 9.3.

```
For (b = 0; b < B; b++)
   Pick an unsatisfied clause C
   For each variable x in clause C
      breakcount(b) := min_{x∈C} break(x)
```

Figure 9.3: Algorithmic structure of the evaluation function ev_2 within BCOi.

Figure 9.3 shows that in order to utilize variable $break$ one first needs to pick a clause. Namely, WalkSAT concentrates the search towards variables that belong to unsatisfied clauses. After the clause is chosen uniformly at random, the procedure determines a minimal value of $break(x)$ for each of the three variables from the clause. The quality of the reported solution of each bee is evaluated with $\min_{x \in C} break(x)$. Consequently, a bee with the minimal value of $breakcount$ is marked as the best one. If all the bees have reached the solution of the same quality, each bee remains loyal to its solution.

It is worth noting that within the evaluation process we may also consider other parameters such as $make$ or $score$ or their combination. Because of time constraints, we have not investigated other evaluation functions. In addition, possible hybrids remain the topic of the future challenges.

9.4 Empirical study of randBCOi

The experimental study of randBCOi has been conducted for two problem-size sets: uf50-218 and uf100-430. We begin with observations for the hard problem set (uf100-430) and utilize an average number of unsatisfied clauses $\overline{n_{uns}}$ as the performance measure. The goal is to gather knowledge about the success of the BCO framework giving the same stopping criteria as in Section 9.2.1 (MAXFLIPS=10^6). The results of randBCOi for uf100-430 show that the algorithm requires large amount of elementary transformations (flips), and consequently $\overline{n_{flip}}$ to find a satisfactory assignment of a formula. Due to large amount of comparisons we present graphics for ($\overline{n_{uns}}$) averaged over set of 100 instances from uf100-430, i.e., for $N_{uns} = \sum_{i=1}^{100} \overline{n_{unsi}}$. Study of randBCOi for uf50-218 considers a performance measure $N_{flip} = \sum_{j=1}^{100} \overline{n_{flip_j}}$.

9.4.1 Case study: randBCOi for uf100-430

The graphics of results in Figs. 9.4 and 9.5 demonstrate levels of success of the corresponding BCOi instances. The graphics were generated in form of matrix-plots. The legend separates intervals of results produced by different configurations of the BCOi instances. The color fields indicate half-open intervals. The blue colour denotes the best configurations of quantitative BCOi parameters B and NC, i.e., configurations that

have produced the lowest N_{uns}. For example, the best results fall into the interval $[0, 1)$. Some implementations were quite unsuccessful, such as p^3 in Fig. 9.4. Here, the legend shows where the minimal value of N_{uns} belongs to (the interval $[13, 14)$). Therefore, for the particular parameter's configuration only results that were generated during the experiments are indicated.

In Fig. 9.4 we compare class I loyalty functions. The matrix-plots show that $p^{3,n_{it}}$ exhibits the worst performance. Moreover, the visual inspection of the matrix-plots reveals that loyalty functions p^2 and p^8 have similar influence on the performance of the BCOi solver and exhibit the best performance in its class. They produce $N_{uns} \in [0, 1)$ on complete domain of the parameter B, and different values of NCT. In particular, if $B = 2$ then the best results are achieved if $NCT = 1$. As population-size is growing, so does the values of NCT that generate good quality solutions. Namely, for $B = 5$ the best set of configuration is $NCT \in \{1, \ldots, 10\}$. The p^1 exhibits a similar behavior, only that if $B = 5$, then $NCT \in \{1, \ldots, 8\}$. Regrading the worst case, it is shown that the standalone heuristic $Rand$ (if $B = 1$) generates the worst quality solutions, as $N_{uns} \in [19, 20]$. Loyalty function p^1 produces the worst quality of solutions ($N_{uns} \in [15, 16]$) for $B > 1, NCT > 60$, while p^2 and p^8 generate $N_{uns} \in [14, 15]$ if $B > 1$ and for particular values of $NCT > 75$. All tests reported the maximal number of flips.

The results of class II functions are presented in Fig. 9.5. Due to large similarities between $p^{5,u}$ and $p^{9,u}$ we provide matrix-plots for $p^{5,u}$. Similarly, we omit matrix-

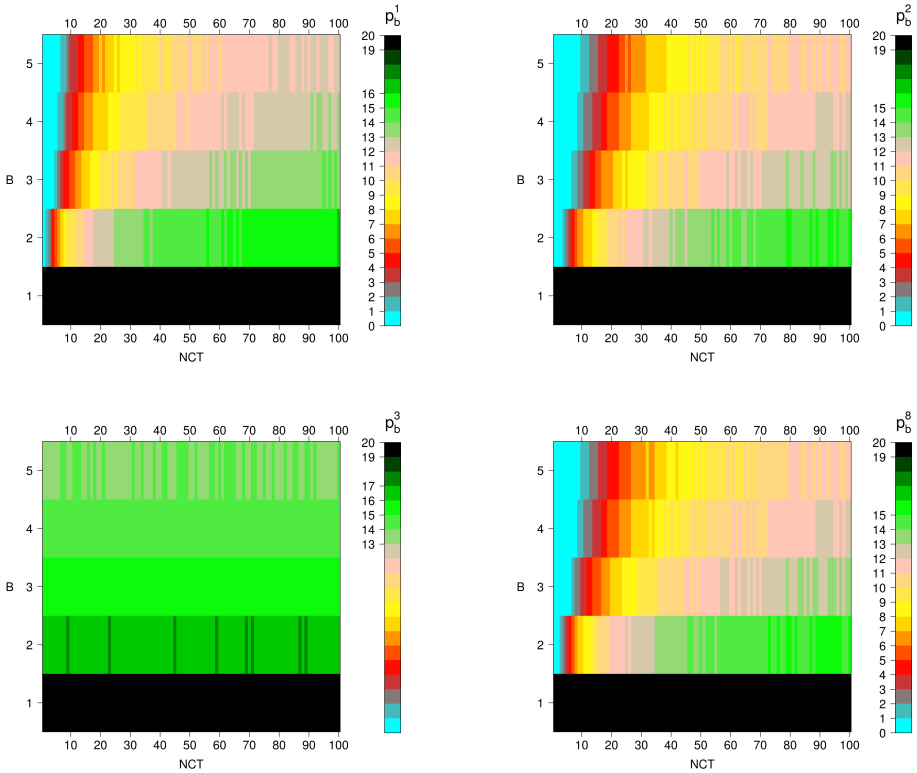


Figure 9.4: Matrix-plots of average number of unsatisfied clauses generated by BCO with the Class I functions: $p^{1,2,8}$ and $p^{3,n_{iter}}$.

plots for $p^{4,u}$ because the graphic matches to results of $p^{6,u}$. From the figure we may conclude that the impact of parameter NC was not significant in case of $p^{5,u}$ ($p^{9,u}$ correspondingly). The largest impact of NC is for $p^{0,u}$, where the quality decreases as the values of NC increase. Thus, smaller values of NC generate high quality results. The same conclusion might be drawn for $p^{7,u}$. Founded on interval coloring, we have observed that the randBCOi solver performs well for small NCT. However, the legends' fields do not indicate concrete values of N_{uns} or indicate differences in computational coast that would split the ties when two BCOi instances generated similar response values. To distinguish between the colored fields we need to consider other information presented in Table 9.6. The results of experiments, not shown in the graphics of results, are elaborated in the rest of this section.

Table 9.6 represents the best randBCOi configurations under the given conditions of the experiments. The average running time and the average number of flips to solve the corresponding problem instances are also presented. The quality of solution has been determined by the smallest N_{uns}. Based on information in Table 9.6 we confirm that the majority of loyalty functions exhibit the best performance for $NCT = 1$. The randBCOi solver improves the workings of an underlying heuristic Rand if we employ backward pass after only one transformation (flip). In particular and according to the quality of solutions, p^2 and p^8 report the best result, i.e., $Nuns = 0.08$. In the text to follow we describe other results not illustrated in the figures.

Loyalty function $p^{5,u}$ shows similar behavior for values of $NC = \{10, 20, \ldots, 60\}$. Moreover, for $NC = 30$ the best performance ($N_{uns} = 0.11$) is reported for $B = 4$ and $NCT = 2$. For the rest the best performance ($N_{uns} = \{0.10, 0.11\}$) is reported for $B = 3$ and $NCT = 1$. In Table 9.6 we show the configuration that has produced $N_{uns} = 0.10$.

Loyalty functions $p^{6,u}$ and $p^{5,u}$ have exhibited a high robustness to changes of parameter NC. However, the differences between reported N_{uns} of the two functions are distinguishable. Namely, the best result of $p^{6,u}$ has the same quality as the result of p^2 and p^8. Therefore, the three loyatly functions have performed equally well.

Loyalty function $p^{7,u}$ shows poor performance compared to the rest class II loyalty functions excluding $p^{3,n_{it}}$. Namely, it performs the best for small NC and the quality

Table 9.6: Results of comparison between Rand and the best configurations of randBCOi for SATLIB problem-set uf100-430. Optimality criterion is N_{uns}.

Lp	B	NC	NCT	Av.Time[s]	N_{flip}	N_{uns}
Rand	-	-	-	0.01728	98771.18	19.80
$p^{0,u}$	5	10	1	0.02742	57515.50	0.18
p^1	3	-	1	0.04825	60496.26	0.11
$\boldsymbol{p^2}$	**2**	-	**1**	**0.03696**	**40604.00**	**0.08**
$p^{3,n_{it}}$	5	-	27	0.01767	98910.40	13.84
$p^{4,u}$	3	10	1	0.02780	48246.57	0.10
$p^{5,u}$	3	20	1	0.04595	58746.78	0.10
$p^{6,u}$	3	10	1	0.03244	48957.84	**0.08**
$p^{7,u}$	5	10	1	0.02717	59549.20	0.25
p^8	3	-	2	0.03241	50413.77	**0.08**
$p^{9,u}$	3	10	1	0.04432	56057.73	0.10

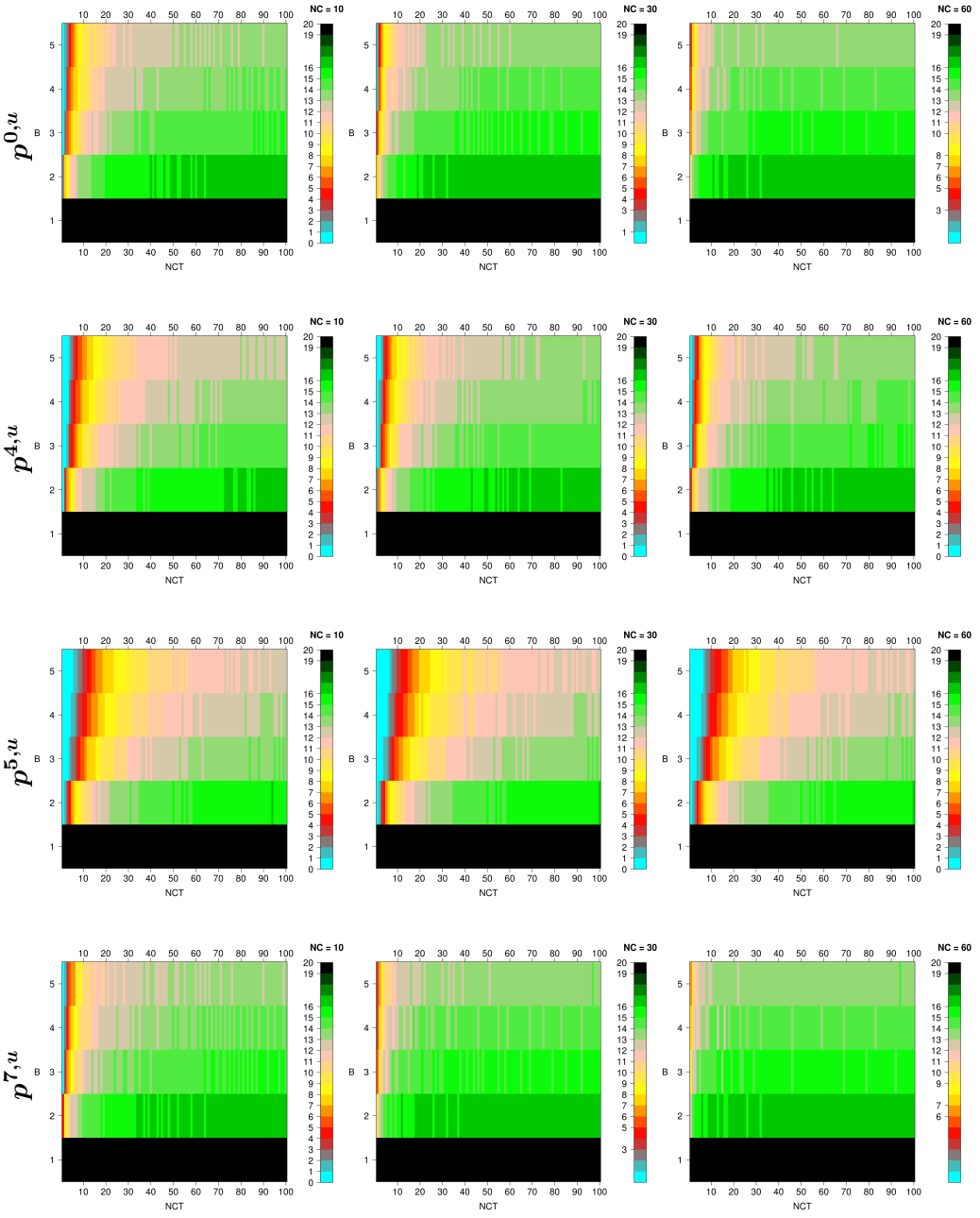


Figure 9.5: Matrix-plots of average number of unsatisfied clauses generated by different BCOi instances and Class II loyalty functions for *uf100* problem-set.

deteriorates significantly as NC increases. In particular, for $NC = 20$ it reports $N_{uns} = 1.39$.

Influence of NC for $p^{9,u}$ is similar to influence of $p^{5,u}$. Namely, the quality of the response values is $N_{uns} = 0.11$ and repeats for $NC = \{20, 30, 40, 50, 60\}$. In the table of results we report parameter configuration that reported $N_{uns} = 0.10$.

The study has produced a large number of results, which we were not able to present

in this thesis. Imperfection of the previous presentation is the lack of information that allows comparison of computational resources between different BCOi configurations. Therefore, we have repeated the experimentation for the smaller size of the problem. In addition, utilizing the same stopping criteria sa previously, the BCOi solver was able to find satisfiable assignments in the majority of the cases. This allowed us to eliminate repotting on N_{uns} and, instead, focus on one performance measure N_{flip}.

9.4.2 Case study: randBCOi for uf50-218

The second study of this section deals with the experimental analysis of the RandBOCi for small-size problem set. The comparison among BCOi parameter's configurations is based on the variable N_{flip}. Stopping criteria is set as in previous studies and according to Section 9.2.1 where we have established MAXFLIPS=10^6. We controlled the $seed$, appointing it to indices of experimental runs.

Results are presented in form of line graphics in Fig. 9.6 and a summary of the best results is given in Table 9.7. For class II, we present graphics for three discrete steps $NC \in \{10, 30, 60\}$. Our observations are, as previously, divided among classes of loyalty functions. Between the loyalty functions of class I, the best results have been reported for $NCT = 1$ and $B = 2$. The most efficient w.r.t. N_{flip} is p^8 that required 1836.62 flips, followed by p^1 with $N_{flip} = 1859.91$ who needed less time in average then p^8. Loyalty function p^2 reported very similar result w.r.t. N_{flip}. Among the results not reported in the graphics we distinguish two cases: $NCT = 1$ and $NCT > 1$. When $NCT = 1$ only $B = \{2, 3\}$ find models of all 100 problem instances for all 100 seeds. Compared against N_{flip} of $B = 2$, larger B reports smaller number of flips only when $NCT \geq 3$ in case of p^1 and when $NCT \geq 5$ for $p^{2,8}$. However, the quality of solution demerits w.r.t. to N_{uns} for $NCT > 40$, regrades of B.

Table 9.7: Results for randBCOi for problem-set uf50-218. Optimality criterion is N_{flip}.

Lp	B	NC	NCT	Av.Time	N_{flip}	N_{uns}
Rand	-	-	-	0.01279	40047.64	0.0074
$p^{0,u}$	3	10	1	0.00551	2760.27	0
$\boldsymbol{p^1}$	**2**	-	**1**	**0.00568**	**1859.91**	0
p^2	2	-	1	0.00805	1904.79	0
$p^{3,n_{it}}$	5	-	5	0.01304	39008.69	0
$p^{4,u}$	3	10	1	0.00680	2485.43	0
$p^{5,u}$	2	30	1	0.00633	1887.52	0
$p^{6,u}$	2	10	1	0.00499	2227.55	0
$p^{7,u}$	3	10	1	0.00505	3069.17	0
$\boldsymbol{p^8}$	**2**	-	**1**	**0.00634**	**1836.62**	0
$p^{9,u}$	2	40	1	0.00535	1893.41	0

Concerning class II, $p^{0,u}$ and $p^{7,u}$ exhibit the weakest performance. Based on comparisons among the best configurations in their class, the two functions require the largest amount of the computational resources. Furthermore, as NC increases the overall efficiency decreases. Similar observation about the influence of the NC parameter is established for other class II loyalty functions. For $p^{5,u}$ and $p^{9,u}$ the parameter has

exhibited the smallest impact . Regrading quantitative BCOi parameters, from graphics we can conclude that configuration $NCT = 1$ generates the best results.

9.4.3 Conclusions for `randBCOi`

We conclude that the BCOi algorithm improves the performance of the underlying solver `Rand` by couple of orders of magnitude w.r.t. computational resources needed to obtain a satisfactory assignment of the corresponding 3-CNF formulas. For uf100-430 the `randBCOi` solver has produced N_{uns} that is 257.5 times smaller than the result of `Rand`. In case of uf50-218, the `randBCOi` solver reports the 21 times smaller N_{flip} until model is found, compared against the underlying solver `Rand`. The best BCOi parameter's configurations are $B = \{2, 3\}$ and $NCT = 1$. The best configurations are reported for class I loyalty functions, however, for some configurations that include class II functions we also observe high quality performance. The most successful for uf100-430 are p^2 and p^8 for uf50-218.

9.5 Experimental study of WalkBCOi

The section is devoted to discussion and presentation of the results reported by the `walkBCOi` solver. The study of analysis of `walkBCOi` was conducted under the same conditions of previous Section 9.4. We conducted the analysis on the uf100-430 problem-set. The results regrading the performance `walkBCOi` are presented in Fig. 9.7 and 9.8 and Tables 9.8. The figures are based on N_{flip} generated on all problem instances and for different BCOi parameter configurations. The goal is to inspect the improvement of quality of solutions by changing the evaluation function. In particular, we test two evaluation functions ev_1 and ev_2, described in Section 9.3.2 with the same set of parameter's domains indicated in Table 9.5. Therefore, we distinguish two case studies with regrade to the type of evaluation function.

9.5.1 Case study: Evaluation function ev_1

The Figure 9.7 provides information about the success of evaluation function ev_1 and ten loyalty functions. The reference case, marked with blue, depicts the performance of WalkSAT, shown as $B = 1$. First conclusion is that `walkBCOi` is more efficient than `randBCOi` for each configuration of the BCOi parameters. However, the reference case reported the smallest N_{flip}. Therefore, `walkBCOi` with ev_1 does not perform better than the `WalkSAT`. To demonstrate the influence of BCOi parametes in Table 9.8 we give a summary of results. We include descriptive statistics for the response value N_{flip} that has produced the best results and the corresponding supplement information such as running time (Av. Time) and N_{uns}. In the table the first row provides response values for the reference case. To understand the influence of parameter B the best results are determined among the parameter configurations for which $B \geq 2$. As in the previous study, the best results w.r.t. N_{flip} are generated for the population-size $B = 2$.

Contrary to the previous study, the overall best loyalty functions are of Class II: $p^{0,u}$m $p^{7,u}$ and $p^{3,n_{it}}$. As shown in Table 9.8 for all $B \geq 2$ configurations, they required the least amount of computation's resource to find a solution of all 3-CNF formulas from the

Table 9.8: Descriptive statistics for `walkBCOi` with ev_1 for problem-set uf100-430. Optimality criterion is N_{flip}.

Lp	B	NC	NCT	Av.Time	N_{flip}	N_{uns}
`WalkSAT`	-	-	-	0.001479	3671.46 ± 3548.48	0
$p^{0,u}$	2	60	91	0.001848	3874.64 ± 3687.65	0
p^{1}	2	-	98	0.002558	5550.34 ± 6169.28	0
p^{2}	2	-	100	0.002278	6848.57 ± 8366.21	0
$p^{3,n_{it}}$	2	-	91	0.001834	3860.89 ± 3699.45	0
$p^{4,u}$	2	50	94	0.002061	4353.94 ± 4374.07	0
$p^{5,u}$	2	10	99	0.002467	5286.15 ± 5794.69	0
$p^{6,u}$	2	60	100	0.002145	4582.22 ± 4702.06	0
$p^{7,u}$	2	60	76	0.001830	3816.29 ± 3625.77	0
p^{8}	2	-		0.002558	6305.75 ± 7198.17	0
$p^{9,u}$	2	10	99	0.002472	5334.06 ± 6019.27	0

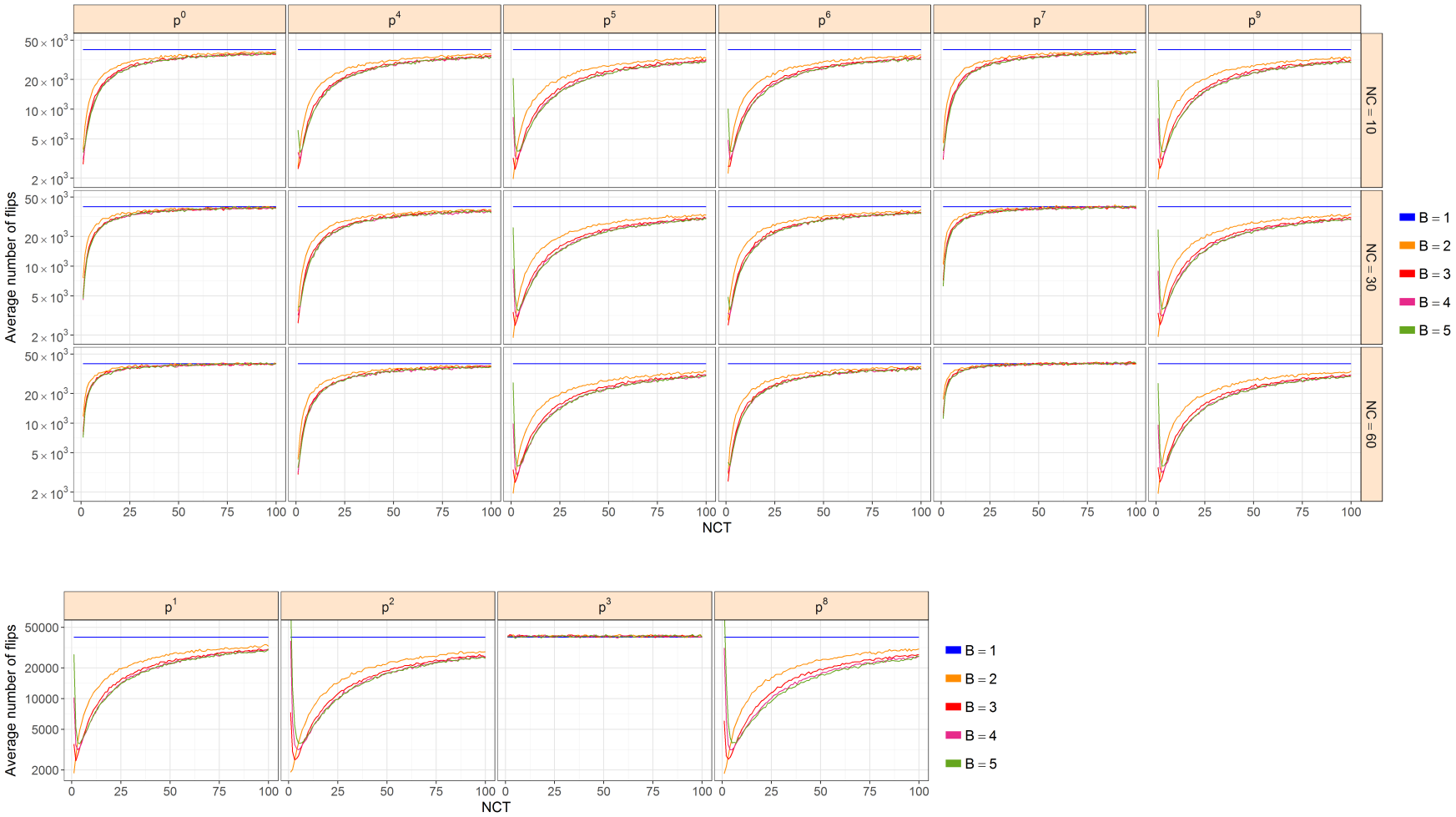


Figure 9.6: Evolution of average number of flips for different `randBCOi` instances. Problem instances belong to class uf50-218. Stopping criteria is MAXFLIPS$=10^6$

family uf100-430. In particular, from Fig. 9.7 we see that $p^{0.u}$ and $p^{7.u}$ show similar line profiles, thus, efficiency for different values of B and NCT. The figure also shows that the parameter NC demonstrated a significant influence on the overall behavior. For larger NC both $p^{0.u}$ and $p^{7.u}$ require smaller number of transformations until finding a model. However, $p^{3.n_{it}}$ was quite unresponsive to changes in quantitative parameters' values. Namely, the results are not practically different. Another conclusion regards efficiency of other loyalty functions. The function p^1 generated better results compared to results reported by p^2 and p^8. However, according to Fig. 9.7 the class I loyalty functions are not as efficient as class II. We conclude that an increase in the number of bees does not improve the performance of the `walkBCOi` solver that calls the ev_1 function.

9.5.2 Case study: Evaluation function ev_2

Figure 9.8 provides information of the success of the evaluation function ev_2. The summary of descriptive data for the best configurations is given Table 9.9. Our first conclusion has been motivated by the research question Q_2 at the beginning of the chapter. Due to considerably smaller number of flips (required until finding a model) and based on the Fig. 9.8 and Table 9.9, the ev_2 function contributes to an increase in the performance of the BCOi algorithm compared to ev_1. Among all loyalty functions $p^{0.u}$, $p^{7.u}$ and $p^{3,n_{it}}$ are the most successful as they solved 3-CNF formulas for small N_{flip}. Furthermore, the corresponding running times in Table 9.9 are similar to the running time of the standalone `WalkSAT` algorithm. In particular, the `WalkSAT` algorithm solved all 3-CNF instances in 0.0015 seconds in average. Significance of the result is found in the potential to implement parallelization strategies. As for the other loyalty functions we have observed that all the configurations required smaller amount of computational resources than in the previous study. The function p^1 shows an improvement in solution quality compared against p^2 and p^8. However, the loyalty functions of class I demonstrated the worst performance.

Table 9.9: Descriptive statistics for `walkBCOi` with ev_2 for problem-set uf100-430. Optimality criterion is N_{flip}.

Lp	B	NC	NCT	Av.Time	N_{flip}	N_{uns}
`WalkSAT`	-	-	-	0.001479	3671.46 ± 3548.48	0
$p^{0,u}$	2	40	100	0.001784	3769.59 ± 3512.48	0
p^1	2	-	96	0.001764	4184.83 ± 4136.13	0
p^2	2	-	100	0.001635	4496.78 ± 4380.62	0
$p^{3,n_{it}}$	2	-	12	0.001488	3781.24 ± 3500.07	0
$p^{4,u}$	2	60	85	0.001883	3944.57 ± 3713.19	0
$p^{5,u}$	2	10	90	0.002017	4216.18 ± 4056.97	0
$p^{6,u}$	2	30	99	0.001680	3990.59 ± 3777.84	0
$p^{7,u}$	2	60	56	0.001445	3728.48 ± 3487.10	0
p^8	2	-	100	0.001500	4380.88 ± 4356.37	0
$p^{9,u}$	2	10	96	0.001798	4226.60 ± 4103.61	0

9.5.3 Conclusions for `walkBCOi`

Total running time of the two considered case studies of `walkBCOi` were close to a week on the cluster specified in Section 7.3.5.2. We emphasize that the runs of experiments were distributed in the following way: each node of the cluster executes series of experiments that correspond to a pair of qualitative parameters. The fastest results were obtained for class I loyalty functions for the both case studies. The slowest BCOi instance in the class II were the $p^{5,u}$ and $p^{9,u}$ loyalty functions. Our conclusion is that the BCOi algorithm improves workings of the random walk technique by hundreds of orders of magnitude. However, implementing more sophisticate modification rules of `WalkSAT` in the BCOi algorithm can produce solutions of a similar quality as the standalone solver. The results of two studies for `walkBCOi` demonstrate that the improvement in the evaluation phase of the algorithm also improves the quality of the overall results.

9.6 Final remarks

Among first empirical results reported about the Schöning's algorithm might be found in [Pre04]. [Bal14a, pg. 26] demonstrate that the implementation of the Schöning's algorithm exhibits weak performance. The result coincides with our observations. Specifically, the reinitialization step in Schöning's algorithm degrades efficiency of the pure random walk procedure and the conclusion is that restarts may be avoided. Another result of our study is that the walksat procedure is efficient for uniform random 3-SAT instances from SATLIB. We suspect that the reason is in the utilization of variable $break$ that exploits knowledge about the consequences of performing a particular step during the search. Moreover, compared to the *novelty* paradigm, walksat does not require the same amount of auxiliary data, therefore, is suitable as set of rules within a population-based meta-heuristics.

BCOi framework improves workings of the random walk solver (`Rand`) by couple of orders of magnitude. In particular, the results indicate that increasing population size leads towards high quality solutions. However, the best results were obtained when population size is small ($B = 2$). Considering qualitative parameters, loyalty decision in `randBCOi` computed by p^2 or p^8 is the most efficient on the considered problem sets. In case of `walkBCOi` the best performing loyalty function is p^3. However, based on results reported by other configurations of the BCOi parameters the proposed design of the BCOi algorithm did not improve performance of the underlying solver. Analysis of evaluation function indicates that there is space for improvement of `walkBCOi` utilizing more information during the evaluation process.

9.7 Chapter summary

The chapter is devoted to the development of the BCOi algorithm for 3-SAT. More precisely:

- We investigate the known solvers for 3-SAT.

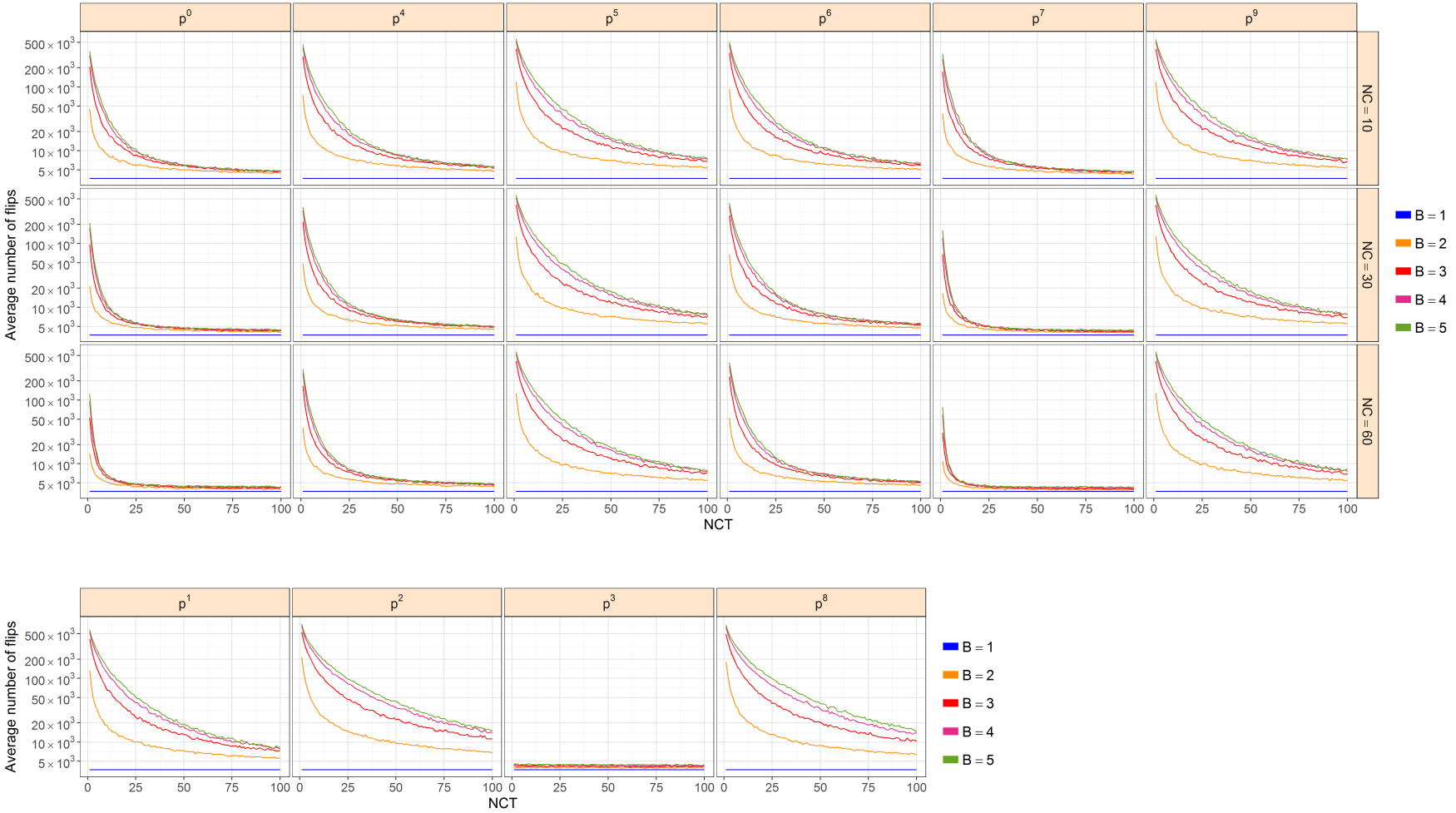


Figure 9.7: Evolution of average number of flips for different `walkBCOi` instances and the evaluation function $ev_1 = numfalse$. Problem instances belong to class uf100–430. Stopping criteria is MAXFLIPS=10^6.

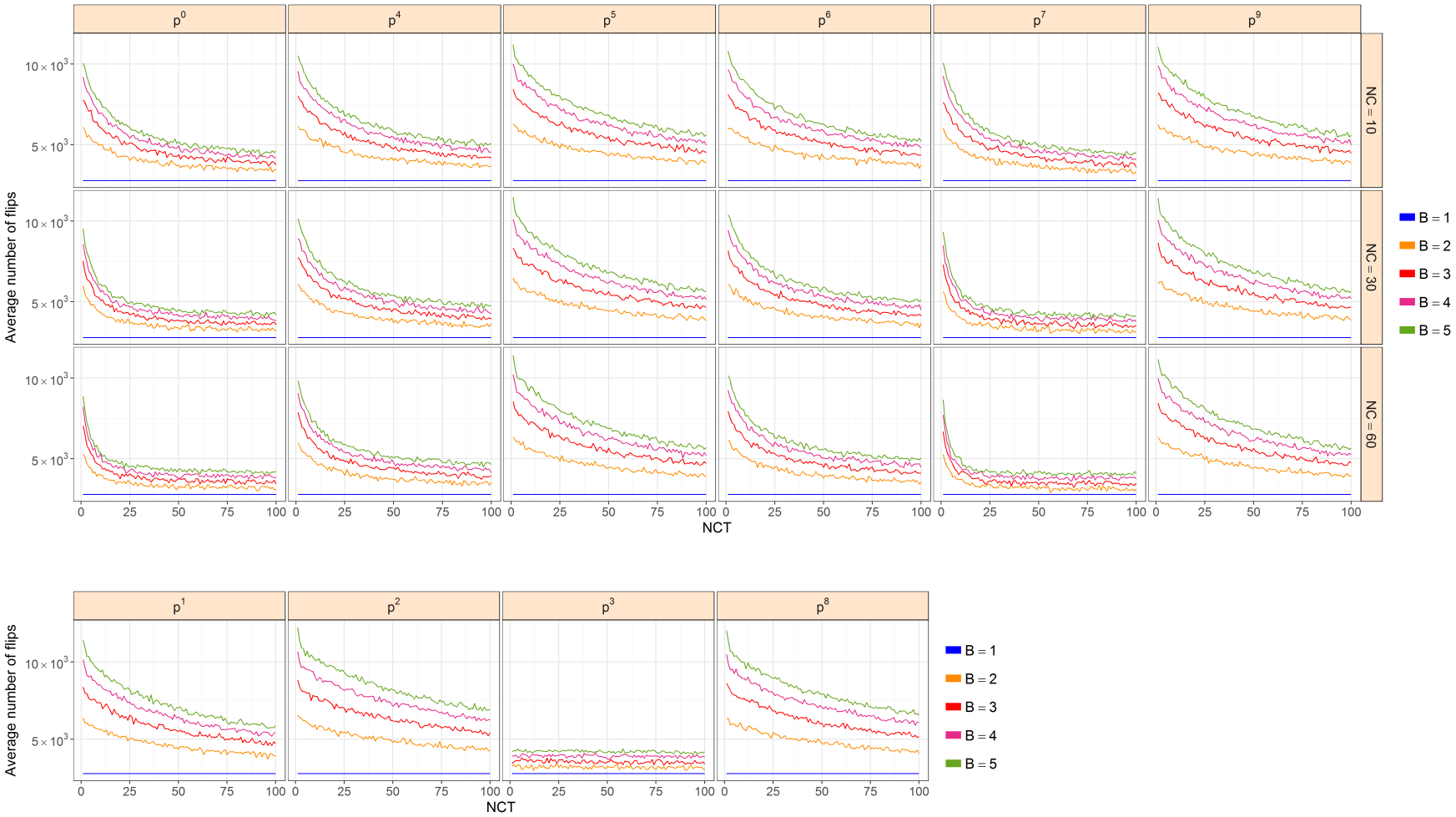


Figure 9.8: Evolution of average number of flips for different `walkBCOi` instances and the evaluation function ev_2 . Problem instances belong to class uf100–430. Stopping criteria is MAXFLIPS=10^6.

- We provide information about the structure of problem instances based on the known properties reported in the literature (e.g., clause-to-variable ratio for 3-SAT).
- We conduct experimental tests to compare several heuristic algorithms for 3-SAT (solvers) and to determine the best candidate for modification rules within the the BCOi algorithm.
- We conduct an extensive analysis of two BCOi algorithms, `randBCOi` and `walkBCOi`. The first has been shown successful compared to `Rand` algorithm. The `walkBCOi` solver has not shown better performance against the standalone solver, however, it produced promising results that might inspire a future work. In particular, we have demonstrated that a change of the evaluation function significantly improves an overall results for all BCOi parameter configurations.

CHAPTER 10

Conclusions and future work

10.1 Concluding remarks

The main goal of this dissertation is the study of the bee colony optimization method and analysis of its characteristics from two perspectives: theoretical and experimental. The research is divided into three directions: theoretical analysis related to asymptotic convergence of the BCO algorithm, implementation of parallelization strategies and empirical analysis of its performance.

Convergence analysis. The central contribution of this thesis is the theoretical verification of BCO convergence towards global optimal solution. Based on recently published tutorials (Section 7.2) we have demonstrated that BCO is well founded and provides a good basis for asymptotic convergence. Assuming that the considered optimization problem has a solution, convergence analysis is related to the question: will the desired optimum be found if the algorithm is given enough time. In particular, we considered two types of convergence, *best-so-far* convergence and the *model* convergence. The first refers to a question whether or not the best-so-far solution converges to some optimal solution, as the number of iterations tends to infinity. The best-so-far convergence is simple to prove for cases with independent iterations. Indeed, the pure random search satisfies its necessary conditions. In case of global knowledge exchange we established necessary notation and conditions that lead BCO towards global optimum. From the tutorials we discovered that efficient meta-heuristic method should concentrate the search more and more towards the most promising areas of the search space exploiting the previous search experience. Therefore, the model convergence analyzes the properties of the algorithm that direct the search process toward the subspaces containing the optimal solution. The proof of the model convergence considers exploration/exploitation tradeoff explicitly into account and only succeeds under parameter assumptions ensuring a proper balance between these two factors. We propose four modification schemes that support establishing such balance. The modification schemes differ by the type of the BCO algorithm and structure of the solution with regard to problem characteristics. As far as further research is concerned with respect to model convergence, convergence speed might be investigated.

Experimental study. Experimental aspect of the thesis contains two approaches: proposing and investigating parallelization strategies for distributed memory processors and experimental study of the BCO algorithm performance. The first approach examines if it is possible to improve the performance of BCOc by exploiting parallelization

concepts. The second approach undertakes various steps in order to explain influence of the BCO algorithmic components on measured outcomes.

Parallelization of BCOc. The BCO method is suitable for parallelization as it operates on a population of solutions by utilizing stochastic, either constructive or improvement, heuristic methods. In particular, we focused on the high-level (coarse-grained) parallelization strategies. The main concept was introduction of Multiple BCO (MBCO), where we vary the values of the BCO parameters and change the stopping criterion at the same time. We consider three strategies for parallelization of MBCO and propose five coarse-grained parallel implementations under the Message Passing Interface (MPI). The first strategy assumes independent execution of various BCO algorithms. Sequential versions of BCOc are executed on different processors and the best solution is collected at the end. Applied to BCOc for $P||C_{max}$, we obtained almost linear speedup for a modest number of engaged processors (≤ 12). At the same time quality of the solution is not degraded significantly (below 3% with respect to the sequential result). The second strategy is related to synchronous cooperative execution of various BCOc instances. We implemented two synchronous cooperative variants on a completely connected homogeneous multiprocessor system, in which processors communicate by exchanging messages. The one involving a less frequent knowledge exchange resulted in better performance for the considered BCOc. Finally, related to the third strategy, we implemented two variants of asynchronous BCO parallelization. The first of them includes a global memory concept, and it is implemented on master-slave multiprocessor architecture. The second, a non-centralized asynchronous execution, is realized on a unidirectional processor ring. On two hard test examples we showed that, while both the synchronous and asynchronous concepts perform well on a modest number of processors, the asynchronous concept outperforms the synchronous one as the number of engaged processors increases. The first contribution of this thesis related to this topic is the successful development of new and efficient distributed memory parallelization strategies for BCO. To the best of our knowledge, we developed the first asynchronous strategies for bee-inspired algorithms, and therefore, they represent the major contribution of this thesis. Future work should focus on exploring OpenMP benefits and designing hybrid implementations.

Empirical study of BCO. The second line of the experimental study in this dissertation is related to sensitivity analysis of BCOc and BCOi algorithms. We studied the behaviour of BCOc by the means of statistical and graphical tools. We established hierarchy diagram 7.1, which led to various research questions and case studies concerning the BCOc performance analysis. Our main contribution concerns structural tuning of the BCOc algorithm for the $P||C_{max}$ scheduling problem. Four candidate heuristics are compared and the best is used in the remainder of the study. In total four BCO parameters were analyzed: B, NC, method of evaluation and loyalty function. The maximal number of bees (population size) is set arbitrary, however, large enough in order to detect any improvements in the solution quality. Values for NC were limited by problem dimension (n). Established statistical results facilitated efforts to detect best structural parameter configuration. Data for each of the configurations are compared for the equivalence of means. Both classical stopping criteria are used, maximal number of iterations and maximum allowed time. The question of suitable value for N_{it}

is thoroughly examined in the first part of Chapter 8. Values for the maxiaml allowed CPU time follows the established line for N_{it} and, furtermore, are defined as function of problem size (n). By means of visual analysis we have conduct parameter tuning. The surface plots help to establish size of effect of method of evaluation. We generate interaction plot, where influence of problem properties, Lp and of ME is observable and their interaction seem significant. Along the study of BCOc we recognized that for some configurations of qualitative BCO parameters, an average solution quality for different pairs (B, NC) does not improve as B increases. Moreover, the response values indicate non-linear surface defined on the configuration space $\mathcal{B}\times\mathcal{NC}$. This suggests interaction between parameters B and NC. However, the interactions were investigated only visually. When setting $N_{it} = 100$ the choice of loyalty function within method $min(ev_1)$ is the most significant and can greatly influence the performance of BCO. Our study showed that on the provided set of problem instances in 50% of the cases the best results were obtained for minimization of ev_1. The results showed that p_b^2 and p_b^8 outperformed other functions. When setting maximal CPU time (changed w.r.t. dimension of the problem) the same conclusion manifested: method $min(ev_1)$ coupled with functions $p_b^{2,8}$ will provide best quality solutions. The main conclusions is the dominance of two loyalty functions p_b^2 and p_b^8 as they were clearly more successful compared to the rest of loyalty functions. High quality solutions were obtained for larger population of bees, that is $B \in [18, 20]$, and when $NC \geq 90$. Combining results of statistical and visual analysis we conclude that configurations $\{min, ev_1, p_b^{2,8}, B \in [18, 20], NC \geq 90\}$ were the most successful. Some additional tests indicate that successful values of NC can be restricted to $[0.9n, n]$, which has yet to be confirmed.

Empirical study of BCOi. We studied the behavior of BCOi on the 3-SAT by utiliznig graphical tools. The study showed that `RandBCOi` improves performance of the `Rand` algorithm on the considered problem set of 3-SAT instances. However, restricted by the maximal number of flips the BCOi algorithm did not succeed to satisfy any of 100 3-CNF formulas. Contrary, `WalkBCOi` that utilizes walksat heuristic showed better performance. Furthermore, we conducted two set of tests in order to compare two evaluation functions. The conclusion is that the performance of `WalkBCOi` is improved once we incorporate knowledge that assists the efforts of the underlying walksat rules. However, the overall success of the BCOi algorithm is not practically better than of `WalkSAT`. The conclusion is that presented BCOi is not suitable as 3-SAT solver and should employ more sophisticated method of evaluation.

10.2 Future work

Besides the fact that BCO method was successfully applied to various combinatorial problems, still enough space is left for further advancement by exploring various information sharing mechanisms and different mechanisms of collaborations. Preliminary work has been reported in [Alz15] where roulette, tournament, rank and disruptive selection have been compared. In dissertation by [Nik15] heterogeneity of bee population was implemented and showed promising results. There is more to contribute to the theoretical aspect of the research, namely, to investigate convergence speed of the BCO algorithm for various optimization problems. Moreover, the empirical verification

of proposed selection schemes in the model convergence section should be conducted and compared against the common implementations.

Regarding experimental analysis of the performance, an imperative directions for future work with BCO algorithms is utilization of the tuning methods reviewed in [Eib11]. Development of exact solutions in the optimization has raised questions about usefulness of heuristic algorithms. Each day new exact algorithms generate optimal solutions for problems that were considered solvable only by use of randomized procedures. The future of heuristic algorithms might lay in the mixture with exact ones. As always John Hooker has provided first insights into this new world in his paper [Hoo13]. A compelling topic for future research is hybridization of BCO with exact methods, MIP based in particular. Furthermore, BCO framework may harness more of techniques of ranking different heuristics with goal to increase the robustness of the BCO algorithm.

Practical aspects of the future work certainly refer to the application of BCO to new and challenging optimization problems: multi-objective optimization, stochastic optimization, problems that require multiple solutions of the equal quality and many others.

Bibliography

[Aar88] Emile Aarts and Jan Korst: *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley & Sons, 1988.

[Abb01] Hussein A. Abbass: MBO: Marriage in honey bees optimization-a Haplometrosis polygynous swarming approach. In *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 1, pp. 207–214. IEEE, 2001.

[Afs07] Abbass Afshar, O. Bozorg Haddad, Miguel A. Mariño, and B.J. Adams: Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. *Journal of the Franklin Institute*, vol. 344(5): pp. 452–462, 2007.

[Alb05] Enrique Alba (ed.): *Parallel Metaheuristics: A New Class of Algorithms*. Wiley Series on Parallel and Distributed Computing. John Wiley & Sons, 2005.

[Alb06] Enrique Alba and Gabriel Luque: Evaluation of parallel metaheuristics. In *Proceedings of the Workshop on Empirical Methods for the Analysis of Algorithms, EMAA'06*, pp. 9–14. Reykjavik, Iceland, September 2006. URL http://www.imada.sdu.dk/~marco/EMAA/Proceedings.html.

[Alb12] Susanne Albers and Matthias Hellwig: Semi-online scheduling revisited. *Theoretical Computer Science*, vol. 443: pp. 1–9, 2012.

[Ale05] Schrijver Alexander: On the History of Combinatorial Optimization (Till 1960). *Handbooks in Operations Research and Management Science: Discrete Optimization*, vol. 12: pp. 1–68, 2005.

[All08] Ali Allahverdi, C.T. Ng, T.C. Edwin Cheng, and Mikhail Y. Kovalyov: A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, vol. 187(3): pp. 985–1032, 2008.

[Alz15] M. Alzaqebah and Salwani Abdullah: Hybrid bee colony optimization for examination timetabling problems. *Computers & Operations Research*, vol. 54: pp. 142–154, 2015.

[And53] R. L. Anderson: Recent Advances in Finding Best Operating Conditions. *Journal of the American Statistical Association*, vol. 48(264): pp. 789–798, 1953.

[Aro98] Sanjeev Arora and Shmuel Safra: Probabilistic Checking of Proofs: A New Characterization of NP. *Journal of the ACM (JACM)*, vol. 45(1): pp. 70–122, 1998.

[Bac97] Thomas Back, David B. Fogel, and Zbigniew Michalewicz (eds.): *Handbook of Evolutionary Computation*. Oxford University Press/IOP Publishing, 1997.

[Bal95] Shumeet Baluja and Rich Caruana: Removing the Genetics from the Standard Genetic Algorithm. In *Proceedings of the XII International Conference on Machine Learning*, pp. 38–46. Tahoce City, California, July 1995.

[Bal99] Dana Harry Ballard: *An Introduction to Natural Computation*. MIT Press, 1999.

[Bal09] Adrian Balint, Daniel Gall, Gregor Kapler, and Robert Retz: Experiment design and administration for computer clusters for SAT-solvers (EDACC). *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 7: pp. 77–82, 2009.

[Bal14a] Adrian Balint: *Engineering stochastic local search for the satisfiability problem*. Ph.D. thesis, Universität Ulm, Fakultät für Ingenieurwissenschaften und Informatik, 2014.

[Bal14b] Adrian Balint, Armin Biere, Andreas Fröhlich, and Uwe Schöning: Improving Implementation of SLS Solvers for SAT and New Heuristics for k-SAT with Long Clauses. In *Theory and Applications of Satisfiability Testing–SAT 2014*, pp. 302–316. Springer, 2014.

[Ban10] Anan Banharnsakun, Tiranee Achalakul, and Booncharoen Sirinaovakul: Artificial bee colony algorithm on distributed environments. In *Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pp. 13–18. IEEE, 2010.

[Bar95] Richard S. Barr, Bruce L. Golden, James P. Kelly, Mauricio G. C. Resende, and William R. Stewart Jr.: Designing and Reporting on Computational Experiments with Heuristic Methods. *Journal of Heuristics*, vol. 1(1): pp. 9–32, 1995.

[Bar11] David Fernández Barrero: *Reliability of performance measures in tree-based Genetic Programming: A study on Koza's computational effort*. Ph.D. thesis, University of Alcalá, 2011.

[Bar15] Blaise Barney: Introduction to Parallel Computing. https://computing.llnl.gov/tutorials/parallel_comp/#DesignPartitioning, 2015.

[Bat08] Roberto Battiti, Mauro Brunato, and Franco Mascia: *Reactive search and intelligent optimization*, vol. 45. Springer, 2008.

[BB04] Thomas Bartz-Beielstein, Konstantinos E. Parsopoulos, and Michael N. Vrahatis: Design and Analysis of Optimization Algorithms Using Computational Statistics. *Applied Numerical Analysis & Computational Mathematics*, vol. 1(2): pp. 413–433, 2004.

[BB06] Thomas Bartz-Beielstein: *Experimental Research in Evolutionary Computation; The New Experimentalism*. Natural Computing Series. Springer, 2006.

[BB13] Thomas Bartz-Beielstein and Mike Preuß: Experimental Analysis of Optimization Algorithms: Tuning and Beyond. In Yossi Borenstein and Alberto Moraglio (eds.), *Theory and Principled Methods for the Design of Metaheuristics*, pp. 205–245. Springer, 2013.

[Bel56] Richard Bellman: Mathematical Aspects of Scheduling Theory. *Journal of the Society for Industrial & Applied Mathematics*, vol. 4(3): pp. 168–205, 1956.

[Ben93] Gerardo Beni and Jing Wang: Swarm Intelligence in Cellular Robotic Systems. In Paolo Dario, Giulio Sandini, and Patrick Aebischer (eds.), *Robots and Biological Systems: Towards a New Bionics? Proceedings of the NATO Advanced Workshop on Robots and Biological System, 1989, Tuscany, Italy*, vol. 102 of *NATO ASI Series*, pp. 703–712. Springer, 1993.

[Ber91] Hugues Bersini and Francisco J. Varela: Hints for adaptive problem solving gleaned from immune networks. In *Parallel Problem Solving from Nature*, vol. 496 of *Lecture Notes in Computer Science*, pp. 343–354. Springer, 1991.

[Bes04] Matthijs Leendert den Besten: *Simple Metaheuristics for Scheduling: An empirical investigation into the application of iterated local search to deterministic scheduling problems with tardiness penalties*. Ph.D. thesis, Technische Universität Darmstadt, Darmstadt, Germany, 2004. Adviser prof. Dr. Wolfgang Bibel.

[Bez92] James C. Bezdek: On the relationship between neural networks, pattern recognition and intelligence. *International Journal of Approximate Reasoning*, vol. 6(2): pp. 85–107, 1992.

[Bil86] Patrick Billingsley: *Convergence of Probability Measures*. John Wiley & Sons, 1986.

[Bir06] Mauro Birattari, Mark Zlochin, and Marco Dorigo: Towards a theory of practice in metaheuristics design: A machine learning perspective. *RAIRO-Theoretical Informatics and Applications*, vol. 40(2): pp. 353–369, 2006.

[Bir09] Mauro Birattari: *Tuning Metaheuristics: A Machine Learning Perspective*, vol. 197 of *Studies in Computational Intelligence*. Springer, 2009.

[Bir10] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle: F-Race and Iterated F-Race: An Overview. In Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss (eds.), *Experimental Methods for the Analysis of Optimization Algorithms*, pp. 311–336. Springer, 2010.

[Bła83] Jacek Błażewicz, Jan Karel Lenstra, and A. H. G. Rinnooy Kan: Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, vol. 5(1): pp. 11–24, 1983.

[Bła07] Jacek Błażewicz, Klaus H. Ecker, Erwin Pesch, Günter Schmidt, and Jan Weglarz: *Handbook on Scheduling: From Theory to Applications*. International Handbook on Information Systems. Springer, 2007.

[Blu03] Christian Blum and Andrea Roli: Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys (CSUR)*, vol. 35(3): pp. 268–308, 2003.

[Blu08] Christian Blum and Andrea Roli: Hybrid Metaheuristics: An Introduction. In *Hybrid Metaheuristics*, vol. 114 of *Studies in Computational Intelligence*, pp. 1–30. Springer, 2008.

[Blu12] Christian Blum, Raymond Chiong, Maurice Clerc, Kenneth De Jong, Zbigniew Michalewicz, Ferrante Neri, and Thomas Weise: Evolutionary Optimization. In *Variants of Evolutionary Algorithms for Real-World Applications*, pp. 1–29. Springer, 2012.

[Boc09] Fayez F Boctor, Jacques Renaud, Angel Ruiz, and Simon Tremblay: Optimal and heuristic solution methods for a multiprocessor machine scheduling problem. *Computers & Operations Research*, vol. 36(10): pp. 2822–2828, 2009.

[Bol93] Massimiliano Bolondi and Massimo Bondanza: *Parallelizzazione di un algoritmo per la risoluzione del problema del commesso viaggiatore*. Master's thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, 1993.

[Bon99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz: *Swarm Intelligence: From Natural to Artificial Systems*. Oxford university press, 1999.

[Bou13] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry: A survey on optimization metaheuristics. *Information Sciences*, vol. 237: pp. 82–117, 2013.

[Box54] George E.P. Box: The Exploration and Exploitation of Response Surfaces: Some General Considerations and Examples. *Biometrics*, vol. 10(1): pp. 16–60, 1954.

[Box05] George E. P. Box, J. Stuart Hunter, and William G. Hunter: *Statistics for Experimenters: Design, Innovation, and Discovery*. John Wiley & Sons, 2nd ed., 2005.

[Boy07] Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi: A tutorial on geometric programming. *Optimization and engineering*, vol. 8(1): pp. 67–127, 2007.

[Brg03] Franc Brglez, Matthias F. Stallmann, and Xiao Yu Li: SATbed: A Configurable Environment for Reliable Performance Experiments with SAT Instance Classes and Algorithms. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing*, pp. 1–15. Santa Margherita Ligure - Portofino, Italy, 2003.

[Bri04] Jack Brimberg, Pierre Hansen, and Nenad Mladenović: Convergence of Variable Neighborhood Search. *Les Cahiers du GERAD*, pp. 1–15, 2004.

[Bro58] Samuel H. Brooks: A Discussion of Random Methods for Seeking Maxima. *Operations research*, vol. 6(2): pp. 244–251, 1958.

[Bro11] Jason Brownlee: *Clever Algorithms: Nature-inspired Programming Recipes*. Jason Brownlee, 2011. www.CleverAlgorithms.com.

[Bru07] Peter Brucker: *Scheduling Algorithms*. Springer-Verlag Berlin Heidelberg, 5th ed., 2007.

[Bru09] Peter Brucker and Sigrid Knust: On the Complexity of Scheduling. In *Introduction to Scheduling*, pp. 1–21. CRC, Taylor & Francis Group, 2009.

[Bus] WikiProject Business: Visual Inspection. Available at: https://en.wikipedia.org/Visual_inspection. 2015.

[But64] A. V. Butrimenko: On the search for optimal routes in changing graphs. *Izv. Akad. Nauk SSSR. Ser. Tekhn. Kibern*, vol. 6, 1964.

[Cal] The Regents of the University of California: What statistical analysis should I use? @ONLINE. http://www.ats.ucla.edu/stat/mult_pkg/whatstat/. Accessed 2015.

[Cam91] Scott Camazine and James Sneyd: A Model of Collective Nectar Source Selection by Honey Bees: Self-organization Through Simple Rules. *Journal of theoretical Biology*, vol. 149(4): pp. 547–571, 1991.

[Cas07] Leandro Nunes de Castro: Fundamentals of natural computing: an overview. *Physics of Life Reviews*, vol. 4(1): pp. 1–36, 2007.

[Čer85] Vladimír Černỳ: Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications*, vol. 45(1): pp. 41–51, 1985.

[Che91] Peter Cheeseman, Bob Kanefsky, and William M. Taylor: Where the Really Hard Problems Are. In *Proceedings of the Twelfth International Joint Conference of Artificial Intellligence, IJCAI-91, 24-30 Aug, 1991, Sydney, Australia*, pp. 331–337. Morgan Kaufmann Publishers, 1991.

[Che99] Bo Chen, Chris N. Potts, and Gerhard J. Woeginger: A Review of Machine Scheduling: Complexity, Algorithms and Approximability. In *Handbook of Combinatorial Optimization*, vol. 3, pp. 21–169. Springer, 1999.

[Che04] Bo Chen: Parallel Scheduling for Early Completion. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, pp. 9.1–9.10. Chapman & Hall/CRC, 2004.

[Che13] Junfeng Chen, Jianjun Ni, and Mingang Hua: Convergence Analysis of a Class of Computational Intelligence Approaches. *Mathematical Problems in Engineering*, vol. 2013: pp. 1–10, 2013.

[Cof78] Edward G. Coffman, Jr., Michael R. Garey, and David S. Johnson: An Application of Bin-Packing to Multiprocessor Scheduling. *SIAM Journal on Computing*, vol. 7(1): pp. 1–17, 1978.

[Cof00] Marie Coffin and Matthew J. Saltzman: Statistical Analysis of Computational Tests of Algorithms and Heuristics. *INFORMS Journal on Computing*, vol. 12(1): pp. 24–44, 2000.

[Coh88] Jacob Cohen: *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 2nd ed., 1988.

[Coh95] Paul R. Cohen: *Empirical Methods for Artificial Intelligence*. The MIT Press, 1995.

[Com11] Georgia Tech College of Computing: The Waggle Dance of the Honeybee [Video file]. Retrieved from https://www.youtube.com/watch?v=bFDGPgXtK-U, 2011.

[Con67] Richard W. Conway, William L. Maxwell, and Louis W. Miller: *Theory of Scheduling*. Addison-Wesley, 1967.

[Coo71] Stephen A. Cook: The Complexity of Theorem-Proving Procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pp. 151–158. ACM, 1971.

[Coo97] Stephen A. Cook and David G. Mitchell: Finding Hard Instances of the Satisfiability Problem. In *Satisfiability Problem: Theory and Applications: DIMACS Workshop, March 11-13, 1996*, vol. 35, pp. 1–17. American Mathematical Society, 1997.

[Cor01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: *Introduction to Algorithms*. MIT press, 2nd ed., 2001.

[Cra05] Teodor Gabriel Crainic and Nourredine Hail: Parallel Metaheuristics Applications. In Enrique Alba (ed.), *Parallel Metaheuristics: A New Class of Algorithms*, pp. 447–494. John Wiley & Sons, 2005.

[Cra07] Teodor Gabriel Crainic and Michel Toulouse: Explicit and Emergent Cooperation Schemes for Search Algorithms. In *International Conference on Learning and Intelligent Optimization*, pp. 95–109. Springer, 2007.

[Cra10] Teodor Gabriel Crainic and Michel Toulouse: Parallel Meta-heuristics. In Michel Gendreau and Jean-Yves Potvin (eds.), *Handbook of metaheuristics*, pp. 497–541. Springer, 2010.

[Cra14] Teodor Gabriel Crainic, Tatjana Davidović, and Dušan Ramljak: Designing Parallel Meta-Heuristic Methods. In Marijana Despotović-Zrakić, Veljko Milutinović, and Aleksandar Belić (eds.), *High Performance and Cloud Computing in Science and Education*, pp. 260–280. 2014.

[Cro12] Carrol Croarkin and Paul Tobias: *NIST/SEMATECH e-Handbook of Statistical Methods*. http://www.itl.nist.gov/div898/handbook/, 2012.

[Cun02] Van-Dat Cung, Simone L. Martins, Celso C. Ribeiro, and Catherine Roucairol: Strategies for the Parallel Implementation of Metaheuristics. In *Essays and surveys in metaheuristics*, vol. 15 of *Operations Research/Computer Science Interfaces Series*, pp. 263–308. Springer, 2002.

[Cza04] Andrew Czarn, Cara MacNish, Kaipillil Vijayan, Berwin Turlach, and Ritu Gupta: Statistical Exploratory Analysis of Genetic Algorithms. *Evolutionary Computation, IEEE Transactions on*, vol. 8(4): pp. 405–421, 2004.

[Dag98] Leonardo Dagum and Ramesh Menon: OpenMP: An Industry Atandard API for Shared-Memory Programming. *IEEE Computational Science and Engineering*, vol. 5(1): pp. 46–55, 1998.

[Dav60] Martin Davis and Hilary Putnam: A Computing Procedure for Quantification Theory. *Journal of the ACM*, vol. 7(3): pp. 201–215, 1960.

[Dav02] Charles S. Davis: *Statistical Methods for the Analysis of Repeated Measurements*. Springer-Verlag New York, 2002.

[Dav06a] Tatjana Davidović: *Rasporedjivanje zadataka na višeprocesorske sisteme primenom metaheuristika*. Ph.D. thesis, Matematički fakultet, Univerzitet u Beogradu, 2006. Advisers prof. Dr. Nenad Mladenović and prof. Dr. Dušan Tošić.

[Dav06b] Tatjana Davidović and Teodor Gabriel Crainic: Benchmark-Problem Instances for Static Scheduling of Task Graphs with Communication Delays on Homogeneous Multiprocessor Systems. *Computers & operations research*, vol. 33(8): pp. 2155–2177, 2006.

[Dav09] Tatjana Davidović, Milica Šelmić, and Dušan Teodorović: Scheduling Independent Tasks: Bee Colony Optimization Approach. In *Proceedings of the 17th Mediterranean Conference on Control and Automation, MED'09*, pp. 1020–1025. Thessaloniki, Greece, 2009.

[Dav11a] Tatjana Davidović, Dušan Ramljak, Milica Šelmić, and Dušan Teodorović: Bee colony optimization for the p-center problem. *Computers & Operations Research*, vol. 38(10): pp. 1367–1376, 2011.

[Dav11b] Tatjana Davidović, Dušan Ramljak, Milica Šelmić, and Dušan Teodorović: MPI Parallelization of Bee Colony Optimization. In *Proceedings of the 1st International Symposium & 10th Balkan Conference on Operational Research*, vol. 2, pp. 193–200. 2011.

[Dav12] Tatjana Davidović, Milica Šelmić, Dušan Teodorović, and Dušan Ramljak: Bee colony optimization for scheduling independent tasks to identical processors. *Journal of Heuristics*, vol. 18(4): pp. 549–569, 2012.

[Dav13] Tatjana Davidović, Tatjana Jakšić, Dušan Ramljak, Milica Šelmić, and Dušan Teodorović: Parallelization Strategies for Bee Colony Optimization Based on Message Passing Communication Protocol. *Optimization*, vol. 62(8): pp. 1113–1142, 2013. Dedicated to BALCOR 2011.

[Dav15a] Tatjana Davidović: Bee Colony Optimization: Recent Developments and Applications. "*Mircea cel Batran*" *Naval Academy Scientific Bulletin*, vol. (Special issue devoted to BALCOR 2015) 18(2): pp. 225–235, 2015.

[Dav15b] Tatjana Davidović, Dušan Teodorović, and Milica Šelmić: Bee Colony Optimization Part I: The Algorithm Overview. *Yugoslav Journal of Operational Research*, vol. 25(1): pp. 33–56, 2015.

[Deb14] Kalyanmoy Deb: Multi-objective Optimization. In *Search methodologies*, pp. 403–449. Springer, 2014.

[Del95] Mauro Dell'Amico and Silvano Martello: Optimal Scheduling of Tasks on Identical Parallel Processors. *ORSA Journal on Computing*, vol. 7(2): pp. 191–200, 1995.

[Del12] Mauro Dell'Amico, Manuel Iori, Silvano Martello, and Michele Monaci: A note on exact and heuristic algorithms for the identical parallel machine scheduling problem. *Journal of Heuristics*, vol. 18(6): pp. 939–942, 2012.

[Der09] Ulrich Derigs (ed.): *Optimization and Operations Research*, vol. 1. EOLSS, 2009.

[Der11] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, vol. 1(1): pp. 3–18, 2011.

[Dim11] Branka Dimitrijević, Dušan Teodorović, Vladimir Simić, and Milica Šelmić: Bee Colony Optimization Approach to Solving the Anticovering Location Problem. *Journal of Computing in Civil Engineering*, vol. 26(6): pp. 759–768, 2011.

[DJ75] Kenneth Alan De Jong: *Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. thesis, University of Michigan, 1975.

[Dor92] Marco Dorigo: *Optimization, Learning and Natural Algorithms*. Ph.D. thesis, 1992.

[Dor99] M. Dorigo and G. Di Caro: Ant Colony Optimization: A New Meta-Heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation, 6-9 July, 1999, Washington, DC, USA*, vol. 2, pp. 1470–1477. IEEE, 1999.

[Dor04] Marco Dorigo and Thomas Stützle: *Ant Colony Optimization*. the MIT press, 2004.

[Dor05] Marco Dorigo and Christian Blum: Ant colony optimization theory: A survey. *Theoretical computer science*, vol. 344: pp. 243–278, 2005.

[Dor07] Marco Dorigo and Mauro Birattari: Swarm intelligence. *Scholarpedia*, vol. 2(9): p. 1462, 2007. URL http://www.scholarpedia.org/Swarm_intelligence.

[Dor10] Marco Dorigo and Thomas Stützle: Ant Colony Optimization: Overview and Recent Advances. In *Handbook of Metaheuristics*, vol. 146 of *International Series in Operations Research & Management Science*, pp. 227–263. Springer, 2010.

[Dri05] Habiba Drias, Souhila Sadeg, and Safa Yahi: Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem. In *Computational Intelligence and Bioinspired Systems*, vol. 3512 of *Lecture Notes in Computer Science*, pp. 318–325. Springer, 2005.

[Dun55] Charles W. Dunnett: A multiple comparison procedure for comparing several treatments with a control. *Journal of the American Statistical Association*, vol. 50(272): pp. 1096–1121, 1955.

[Dut04] Joydeep Dutta: Optimization Theory-A Modern face of Applied Mathematics. *Directions*, vol. 6(3): pp. 19–24, 2004.

[Eda08] Praveen Edara, Milica Šelmić, and Dušan Teodorović: Heuristic solution algorithms for a traffic sensor optimization problem. *INFORMS*, pp. 12–15, 2008.

[Eib02] Agoston E Eiben and Márk Jelasity: A Critical Note on Experimental Research Methodology in EC. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'2002)*, vol. 1, pp. 582–587. 2002.

[Eib11] Agoston E. Eiben and Selmar K. Smit: Parameter Tuning for Configuring and Analyzing Evolutionary Algorithms. *Swarm and Evolutionary Computation*, vol. 1(1): pp. 19–31, 2011.

[EO13] Jawad A. El-Omari: *Efficient Learning Methods to Tune Algorithm Parameters*. Ph.D. thesis, University of Warwick, 2013. Adviser prof. Dr. Juergen Branke.

[Fag09] Morten W. Fagerland and Leiv Sandvik: The Wilcoxon–Mann–Whitney test under scrutiny. *Statistics in medicine*, vol. 28(10): p. 1487, 2009.

[Fal96] Emanuel Falkenauer: A hybrid grouping genetic algorithm for bin packing. *Journal of heuristics*, vol. 2(1): pp. 5–30, 1996.

[FB07] David Fernández-Baca and Balaji Venkatachalam: Sensitivity analysis in combinatorial optimization. In Teofilo F. Gonzalez (ed.), *Handbook of Approximation Algorithms and Metaheuristics*, Computer and Information Science Series, pp. 30.1–30.17. Chapman&Hall/CRC, 2007.

[Feo95] Thomas A. Feo and Mauricio G. C. Resende: Greedy Randomized Adaptive Search Procedures. *Journal of global optimization*, vol. 6(2): pp. 109–133, 1995.

[Flo09] Christodoulos A. Floudas and Panos M. Pardalos (eds.): *Encyclopedia of Optimization*. Springer, 2nd ed., 2009.

[Fod12] János Fodor, Ryszard Klempous, and Carmen Paz Suárez Araujo (eds.): *Recent Advances in Intelligent Engineering Systems*, vol. 378 of *Studies in Computational Intelligence*. Springer, 2012.

[FP10] Luis Fanjul-Peyro and Rubén Ruiz: Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, vol. 207(1): pp. 55–69, 2010.

[Fra94] Jeremy Frank: A study of genetic algorithms to find approximate solutions to hard 3CNF problems. In *Golden West International Conference on Artificial Intelligence*. 1994.

[Fra10] Eitan Frachtenberg and Uwe Schwiegelshohn: Preface. In *15th Internation Workshop, JSSPP 2010, Job Scheduling Strategies for Parallel Processing*, pp. V–VII. 2010.

[Gal91] Efim A. Galperin: Problem-method classification in optimization and control. *Computers & Mathematics with Applications*, vol. 21(6): pp. 1–6, 1991.

[Gal97] Tomas Gal and Harvey J. Greenberg (eds.): *Advances in Sensitivity Analysis and Parametric Programming*. International Series in Operations Research & Management Science. Springer, 1997.

[Gar79] Michael R. Garey and David S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of books in the mathematical sciences. W. H. Freeman and Company, 1979.

[Gas13] Saul I. Gass and Michael C. Fu (eds.): *Encyclopedia of operations research and management science*. Springer, New York, 2013.

[Gen93] Ian P. Gent and Toby Walsh: Towards an understanding of hill-climbing procedures for SAT. In *Proceedings of AAAI'93*, vol. 93, pp. 28–33. MIT Press, 1993.

[Gen10a] Michel Gendreau and Jean-Yves Potvin: *Handbook of metaheuristics*, vol. 2. Springer, 2010.

[Gen10b] Michel Gendreau and Jean-Yves Potvin: Tabu search. In *Handbook of Metaheuristics*, pp. 41–59. Springer, 2010.

[GG12] Esperanza Garcia-Gonzalo and Juan Luis Fernandez-Martinez: A brief historical review of particle swarm optimization (PSO). *Journal of Bioinformatics and Intelligent Control*, vol. 1(1): pp. 3–16, 2012.

[Gla94] Celia A. Glass, Chris N. Potts, and P. Shade: Unrelated parallel machine scheduling using local search. *Mathematical and Computer Modelling*, vol. 20(2): pp. 41–52, 1994.

[Glo77] Fred Glover: Heuristics for integer programming using surrogate constraints. *Decision Sciences*, vol. 8(1): pp. 156–166, 1977.

[Glo86] Fred Glover: Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, vol. 13(5): pp. 533–549, 1986.

[Glo89] Fred Glover: Tabu search-part I. *ORSA Journal on computing*, vol. 1(3): pp. 190–206, 1989.

[Glo97] Fred Glover and Manuel Laguna: Tabu search. *Kluwer Academic Publishers*, 1997.

[Gol88] Olivier Goldschmidt and Dorit S. Hochbaum: Polynomial algorithm for the k-cut problem. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pp. 444–451. IEEE, 1988.

[Gol89] David Edward Goldberg *et al.*: *Genetic algorithms in search, optimization, and machine learning*, vol. 412. Addison-wesley Reading Menlo Park, 1989.

[Gol99] Oded Goldreich: *Introduction to Complexity Theory - Lecture Notes*. Weizmann Institute of Science, Israel, 1999.

[Gon07] Teofilo F. Gonzalez (ed.): *Handbook of approximation algorithms and metaheuristics*. Computer and Information Science Series. Chapman&Hall/CRC, 2007.

[Got02] Jens Gottlieb, Elena Marchiori, and Claudio Rossi: Evolutionary algorithms for the satisfiability problem. *Evolutionary Computation*, vol. 10(1): pp. 35–50, 2002.

[Gou75] James L. Gould: Honey bee recruitment: the dance-language controversy. *Science*, vol. 189(4204): pp. 685–693, 1975.

[Gra66] Ronald L. Graham: Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, vol. 45(9): pp. 1563–1581, 1966.

[Gra69] Ronald L. Graham: Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, vol. 17(2): pp. 416–429, 1969.

[Gra79] Ronald L. Graham, Eugene L. Lawler, Jan Karel Lenstra, and A.H.G. Rinnooy Kan: Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, vol. 5: pp. 287–326, 1979.

[Gra94] Vincent Granville, Mirko Krivánek, and J-P Rasson: Simulated annealing: A proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16(6): pp. 652–656, 1994.

[Gra12] Boris Granovskiy, Tanya Latty, Michael Duncan, David JT Sumpter, and Madeleine Beekman: How dancing honey bees keep track of changes: the role of inspector bees. *Behavioral Ecology*, vol. 23(3): pp. 588–596, 2012.

[Gre86] John J. Grefenstette: Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on systems, man, and cybernetics*, vol. 16(1): pp. 122–128, 1986.

[Grö12] Martin Grötschel: *Optimization Stories*, vol. 1. Dt. Mathematiker-Vereinigung, 2012.

[Gu92] Jun Gu: Efficient local search for very large-scale satisfiability problems. *SIGART Bulletin*, vol. 3(1): pp. 8–12, 1992.

[Gut00] Walter J. Gutjahr: A graph-based ant system and its convergence. *Future generation computer systems*, vol. 16(8): pp. 873–888, 2000.

[Gut02] Walter J. Gutjahr: ACO algorithms with guaranteed convergence to the optimal solution. *Information processing letters*, vol. 82(3): pp. 145–153, 2002.

[Gut09] Walter J. Gutjahr: Convergence analysis of metaheuristics. In *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, pp. 159–187. Springer, 2009.

[Gut10] Walter J. Gutjahr: Stochastic search in metaheuristics. In *Handbook of Metaheuristics*, pp. 573–597. Springer, 2010.

[Gut11] Walter J. Gutjahr: Ant Colony Optimization: Recent Developments in Theoretical Analysis. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, pp. 225–254, 2011.

[Hai10] Tina Haidborn: Dancing with bees. *Max Plank Research*, vol. 2: pp. 75–80, 2010.

[Haj88] Bruce Hajek: Cooling schedules for optimal annealing. *Mathematics of operations research*, vol. 13(2): pp. 311–329, 1988.

[Han86] Pierre Hansen: The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming. In *Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy*, pp. 70–145. 1986.

[Han90] Pierre Hansen and Brigitte Jaumard: Algorithms for the maximum satisfiability problem. *Computing*, vol. 44(4): pp. 279–303, 1990.

[Han01] Saïd Hanafi: On the convergence of tabu search. *Journal of Heuristics*, vol. 7(1): pp. 47–58, 2001.

[Han03] Pierre Hansen and Nenad Mladenović: Variable neighborhood search. In F. Glower and G. Kochenagen (eds.), *Handbook of Metaheuristics*, pp. 145–184. Springer, 2003.

[Han10a] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros: Real-parameter black-box optimization benchmarking 2010: Experimental setup. Tech. Rep. RR-7215, INRIA, 2010.

[Han10b] Pierre Hansen, Nenad Mladenović, Jack Brimberg, and José A. Moreno Pérez: Variable neighborhood search. In *Handbook of Metaheuristics*, pp. 61–86. Springer, 2010.

[Han14] Pierre Hansen and Nenad Mladenović: Variable neighborhood search. In *Search methodologies*, pp. 313–337. Springer, 2014.

[Har90] Richard F. Hartl: A Global Convergence Proof for a Class of Genetic Algorithms. Tech. rep., University of Vienna, Institute of Management, 1990.

[Hau04] Randy L. Haupt and Sue Ellen Haupt: *Practical Genetic Algorithms*. John Wiley & Sons, 2004.

[Hed04] Shawn Hedman: *A First Course in Logic*. Oxford University Press, 2004.

[Hol92] John Henry Holland: *Adaptation in Natural and Artificial Aystems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, 1992.

[Hol99] Myles Hollander, Douglas A. Wolfe, and Eric Chicken: *Nonparametric statistical methods*. 2. John Wiley & Sons, 1999.

[Hoo94] John N. Hooker: Needed: An empirical science of algorithms. *Operations Research*, vol. 42(2): pp. 201–212, 1994.

[Hoo95] John N. Hooker: Testing heuristics: We have it all wrong. *Journal of Heuristics*, vol. 1(1): pp. 33–42, 1995.

[Hoo98a] Holger H. Hoos: *Stochastic Local Search – Methods, Models, Applications*. Ph.D. thesis, Technishen Universität Darmstadt, 1998. Adviser prof. Dr. Wolfgang Bibel.

[Hoo98b] Holger H. Hoos and Thomas Stützle: Evaluating Las Vegas Algorithms: Pitfalls and Remedies. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (UAI-98)*, pp. 238–245. Morgan Kaufmann Publishers, 1998.

[Hoo99] Holger H. Hoos and Thomas Stützle: Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. *Artificial Intelligence*, vol. 112(1): pp. 213–232, 1999.

[Hoo00a] Holger H. Hoos and Thomas Stützle: Local search algorithms for SAT: An empirical evaluation. *Journal of Automated Reasoning*, vol. 24(4): pp. 421–481, 2000.

[Hoo00b] Holger H. Hoos and Thomas Stützle: SATLIB: An Online Resource for Research on SAT. In *SAT2000: Highlights of Satisfiability Research in the Year 2000 (Frontiers in Artificial Intelligence and Applications)*, pp. 283–292. 2000.

[Hoo05] Holger H. Hoos and Thomas Stützle: *Stochastic local search: Foundations & applications*. Elsevier, 2005.

[Hoo07] Holger H. Hoos and Thomas Stützle: Empirical Analysis of Randomized Algorithms. *Handbook of Approximation Algorithms and Metaheuristics*, 2007.

[Hoo09] Holger H. Hoos: A Bootstrap Approach to Analysing the Scaling of Empirical Run-time Data with Problem Size. Tech. Rep. TR-2009-16, University of British Columbia Available, 2009.

[Hoo11] Holger H. Hoos: Automated algorithm configuration and parameter tuning. In Youssef Hamadi, Eric Monfroy, and Frédéric Saubion (eds.), *Autonomous search*, pp. 37–71. Springer, 2011.

[Hoo13] John N. Hooker: Toward unification of exact and heuristic optimization methods. *International Transactions in Operational Research*, vol. 22(1): pp. 19–48, 2013.

[How10] David Howell: *Statistical Methods for Psychology*. Cengage Learning, 2010.

[Hsu96] Jason Hsu: *Multiple comparisons: theory and methods*. CRC Press, 1996.

[Hut07] Frank Hutter, Holger H. Hoos, and Thomas Stützle: Automatic algorithm configuration based on local search. In *AAAI*, pp. 1152–1157. 2007.

[Hut10a] Frank Hutter, Thomas Bartz-Beielstein, Holger H. Hoos, Kevin Leyton-Brown, and Kevin P. Murphy: Sequential model-based parameter optimization: An experimental investigation of automated and interactive approaches. In *Experimental Methods for the Analysis of Optimization Algorithms*, pp. 363–414. Springer, 2010.

[Hut10b] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown: Sequential model-based optimization for general algorithm configuration (extended version). Tech. Rep. TR-2010-10, University of British Columbia, Computer Science, 2010.

[Hut11] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown: Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization (LION'11)*, pp. 507–523. 2011.

[Jü11] Jüppsche and Wikimedia Commons: Bee dance. https://commons.wikimedia.org/wiki/File:Bee_dance.svg, 2011. File: Bee_dance.svg.

[Jac04a] Sheldon H. Jacobson and Enver Yücesan: Analyzing the performance of generalized hill climbing algorithms. *Journal of Heuristics*, vol. 10(4): pp. 387–405, 2004.

[Jac04b] Sheldon H. Jacobson and Enver Yücesan: Global Optimization Performance Measures for Generalized Hill Climbing Algorithms. *Journal of Global Optimization*, vol. 29(2): pp. 173–190, 2004.

[Jam95] Kennedy James and Eberhart Russell: Particle swarm optimization. In *Proceedings of 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948. 1995.

[Jia07] M. Jiang, Y.P. Luo, and S.Y. Yang: Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, vol. 102(1): pp. 8–16, 2007.

[JK14a] Tatjana Jakšić Krüger: O konvergenciji metaheurističke metode optimizacija kolonijom pčela. In *Zbornik IV Simpozijuma "Matematika i primene"*, pp. 176–188. Matematički fakultet Univerziteta u Beogradu, 2014.

[JK14b] Tatjana Jakšić Krüger and Tatjana Davidović: Model Convergence Properties of the Constructive Bee Colony Optimization Algorithm. In *Proceedings of 41th Symposium on Operational Research, SYM-OP-IS 2014*, pp. 340–345. 2014.

[JK16a] Tatjana Jakšić Krüger and Tatjana Davidović: Empirical Analysis of the Bee Colony Optimization Method on 3-SAT problem. pp. 297–301. 2016.

[JK16b] Tatjana Jakšić Krüger, Tatjana Davidović, Dušan Teodorović, and Milica Šelmić: The Bee Colony Optimization Algorithm and its Convergence. *International Journal of Bio-Inspired Computation*, vol. 8(5): pp. 340–354, 2016.

[JK16c] Tatjana Jakšić Krüger and Davidović, Tatjana: Sensitivity analysis of the Bee Colony Optimization Algorithm. pp. 64–80. 2016.

[Joh74] David S. Johnson, Alan Demers, Jeffrey D. Ullman, Michael R. Garey, and Ronald L. Graham: Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, vol. 3(4): pp. 299–325, 1974.

[Joh02a] Alan W. Johnson and Sheldon H. Jacobson: On the convergence of generalized hill climbing algorithms. *Discrete Applied Mathematics*, vol. 119(1): pp. 37–57, 2002.

[Joh02b] David S. Johnson: A theoretician's guide to the experimental analysis of algorithms. *Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges*, vol. 59: pp. 215–250, 2002.

[Joh06] Michael P. Johnson, Bryan Norman, and Nicola Secomandi: *Tutorials in Operations Research*. Institute for Operations Research and the Management Sciences (INFORMS), 2006.

[Jov05] Dejan Jovanović, Nenad Mladenović, and Zoran Ognjanović: Variable neighborhood search for the probabilistic satisfiability problem. In *Proceedings of the 6th Metaheuristics International Conference (MIC2005), Vienna, Austria, August 22-26*, pp. 557–562. 2005.

[Jov09] Raka Jovanović, Milan Tuba, and Dana Simian: Analysis of Parallel Implementations of the Ant Colony Optimization Applied to the Minimum Weight Vertex Cover Problem. In *Proceedings of the 9th WSEAS International Conference on Simulation, modelling and optimization*, pp. 254–259. WSEAS, 2009.

[Kam09] Krithivasan Kamala: *Introduction to formal languages, automata theory and computation*. Pearson Education India, 2009.

[Kao08] Ming-Yang Kao (ed.): *Encyclopedia of Algorithms*. Springer, 2008.

[Kar63] Dean C. Karnopp: Random search techniques for optimization problems. *Automatica*, vol. 1(2): pp. 111–121, 1963.

[Kar72] Richard M. Karp: Reducibility among Combinatorial Problems. *Complexity of Computer Computations*, pp. 85–103, 1972.

[Kar93] Alan F. Karr: *Probability*. Springer Science + Business Media, New York, 1993.

[Kar05] Dervis Karaboga: An Idea Based on Honey Bee Swarm for Numerical Optimization. Tech. rep., 2005.

[Kar07] Dervis Karaboga, Bahriye Akay, and Celal Ozturk: Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In *Modeling decisions for artificial intelligence*, pp. 318–329. Springer, 2007.

[Ken01] James Kennedy, James F Kennedy, and Russell C Eberhart: *Swarm intelligence*. Morgan Kaufmann, 2001.

[Kim13] Hae-Young Kim: Statistical notes for clinical researchers: assessing normal distribution (2) using skewness and kurtosis. *Restorative dentistry & endodontics*, vol. 38(1): pp. 52–54, 2013.

[Kir83] Scott Kirkpatrick, C. D. Gelatt, and Mario P. Vecchi: Optimization by simulated annealing. *Science*, vol. 220(4598): pp. 671–680, 1983.

[Kir07] Roger E. Kirk: *Statistics: an introduction*. Thomson Wadsworth, 2007.

[Kir14] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj: Optimization Techniques: An Overview. In *Multidimensional Particle Swarm Optimization for Machine Learning and Pattern Recognition*, pp. 13–44. Springer, 2014.

[Köt12] Timo Kötzing, Frank Neumann, Heiko Röglin, and Carsten Witt: Theoretical analysis of two ACO approaches for the traveling salesman problem. *Swarm Intelligence*, vol. 6(1): pp. 1–21, 2012.

[Kov13] Nataša Kovač: Bee colony optimization algorithm for the minimum cost berth allocation problem. In *XI Balkan Conference on Operational Research, Balcor*, pp. 245–254. 2013.

[Lai14] Chung Yee Johnny Lai: *Hybrid intelligent optimization techniques and its industrial applications*. Ph.D. thesis, The Hong Kong Polytechnic University, 2014.

[Law76] Eugene L. Lawler: *Combinatorial optimization: networks and matroids*. Courier Dover Publications, 1976.

[Law15] Michael A. Lawrence: ez package. http://github.com/mike-lawrence/ez, 2015.

[LB14] Kevin Leyton-Brown, Holger H. Hoos, Frank Hutter, and Lin Xu: Understanding the empirical hardness of NP-complete problems. *Communications of the ACM*, vol. 57(5): pp. 98–107, 2014.

[Leh06] Erich L. Lehmann and Joseph P. Romano: *Testing statistical hypotheses*. Springer Science & Business Media, 2006.

[Len77] Jan Karel Lenstra, AHG Rinnooy Kan, and Peter Brucker: Complexity of machine scheduling problems. *Annals of discrete mathematics*, vol. 1: pp. 343–362, 1977.

[Les11] Stefan Lessmann, Marco Caserta, and Idel Montalvo Arango: Tuning metaheuristics: A data mining based approach for particle swarm optimization. *Expert Systems with Applications*, vol. 38(10): pp. 12826–12838, 2011.

[Leu04] Jospeh Y-T Leung: Handbook of Scheduling. *Chapmann & Hall/CRC, New York*, 2004.

[Lev11] T. V. Levanova and E. A. Tkachuk: Development of a Bee Colony Optimization Algorithm for the Capacitated Plant Location Problem. In *II International conference, Optimization and applications (OPTIMA-2011)*, pp. 25–09. 2011.

[LI14] Manuel López-Ibáñez and Thomas Stützle: Automatically improving the anytime behaviour of optimisation algorithms. *European Journal of Operational Research*, vol. 235(3): pp. 569–582, 2014.

[Lim09] Chee Peng Lim and Lakhmi C. Jain: Advances in Swarm Intelligence. In *Innovations in Swarm Intelligence*, vol. 248 of *Studies in Computational Intelligence*, pp. 1–7. Springer Berlin Heidelberg, 2009.

[Liu09] Hongbo Liu, Ajith Abraham, and Václav Snásel: Convergence Analysis of Swarm Algorithm. In *NaBIC*, pp. 1714–1719. 2009.

[Lou03] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle: Iterated local search. In *Handbook of metaheuristics*, vol. 57, pp. 320–353. Springer, 2003.

[Lou10] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle: Iterated local search: Framework and applications. In *Handbook of Metaheuristics*, pp. 363–397. Springer, 2010.

[Luč01] Panta Lučić and Dušan Teodorović: Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence. In *Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis*, pp. 441–445. 2001.

[Luč02a] Panta Lučić: *Modeling transportation problems using concepts of swarm intelligence and soft computing*. Ph.D. thesis, Virginia Polytechnic Institute and State University, 2002.

[Luč02b] Panta Lučić and Dušan Teodorović: Transportation modeling: an artificial life approach. In *Proceedings. 14th IEEE International Conference on Tools with Artificial Intelligence, 2002.(ICTAI 2002).*, pp. 216–223. IEEE, 2002.

[Luč03a] Panta Lučić and Dušan Teodorović: Computing with bees: attacking complex transportation engineering problems. *International Journal on Artificial Intelligence Tools*, vol. 12(03): pp. 375–394, 2003.

[Luč03b] Panta Lučić and Dušan Teodorović: Vehicle Routing Problem With Uncertain Demand at Nodes: The Bee System and Fuzzy Logic Approach. In J. L. Verdegay (ed.), *Fuzzy Sets based Heuristics for Optimization*, pp. 67–82. Physica Verlag: Berlin Heidelberg, 2003.

[Lyn06] Inês Lynce and Joao Marques-Silva: SAT in Bioinformatics: Making the Case with Haplotype Inference. In *Theory and Applications of Satisfiability Testing-SAT 2006*, vol. 4121 of *Lecture Notes in Computer Science*, pp. 136–141. Springer, 2006.

[Mak13] Petar Maksimović and Tatjana Davidović: Parameter Calibration in the Bee Colony Optimization Algorithm. In *Proc. 11th Balkan Conf. on Operational Research*, pp. 263–272. BALCOR, 2013.

[Man06] Max Manfrin, Mauro Birattari, Thomas Stützle, and Marco Dorigo: Parallel Ant Colony Optimization for the Traveling Salesman Problem. In *Proc. 5th Int. Workshop on Ant Colony Optimization and Swarm Intelligence (Ants)*, pp. 224–234. Springer, 2006.

[Man09] Vittorio Maniezzo, Thomas Stützle, and Stefan Vos: *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, vol. 10. Springer, 2009.

[Mar73] Joseph Durward Marsh: *Scheduling parallel processors*. Ph.D. thesis, Georgia Institute of Technology, 1973.

[Mar97] Oded Maron and Andrew W Moore: The racing algorithm: Model selection for lazy learners. In *Lazy learning*, pp. 193–225. Springer, 1997.

[Mar05] Leonid Margolin: On the convergence of the cross-entropy method. *Annals of Operations Research*, vol. 134(1): pp. 201–214, 2005.

[Mar07] Goran Marković, Dušan Teodorović, and Vladanka Aćimović-Raspopović: Routing and wavelength assignment in all-optical networks based on the bee colony optimization. *AI Communications*, vol. 20(4): pp. 273–285, 2007.

[McA97] David McAllester, Bart Selman, and Henry Kautz: Evidence for invariants in local search. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, pp. 321–326. AAAI Press, 1997.

[McC87] Charles E. McCulloch: Tests for equality of variances with paired data. *Communications in statistics-theory and methods*, vol. 16(5): pp. 1377–1391, 1987.

[McD14] John H. McDonald: *Handbook of biological statistics*, vol. 3. Sparky House Publishing, Baltimore, Maryland, 2014.

[McG96] Catherine C. McGeoch: Toward an Experimental Method for Algorithm Simulation. *Journal on Computing*, vol. 8(1): pp. 1–15, 1996.

[McN59] Robert McNaughton: Scheduling with deadlines and loss functions. *Management Science*, vol. 6(1): pp. 1–12, 1959.

[Mer06] Stephan Mertens, Marc Mézard, and Riccardo Zecchina: Threshold values of random K-SAT from the cavity method. *Random Structures & Algorithms*, vol. 28(3): pp. 340–373, 2006.

[Méz02a] Marc Mézard, Giorgio Parisi, and Riccardo Zecchina: Analytic and algorithmic solution of random satisfiability problems. *Science*, vol. 297(5582): pp. 812–815, 2002.

[Méz02b] Marc Mézard and Riccardo Zecchina: Random k-satisfiability problem: From an analytic solution to an efficient algorithm. *Physical Review E*, vol. 66: p. 056126, 2002.

[Mic04] Zbigniew Michalewicz and David B. Fogel: *How to solve it: modern heuristics*. Springer, 2004.

[Mid02] Martin Middendorf, Frank Reischle, and Hartmut Schmeck: Multi colony ant algorithms. *Journal of Heuristics*, vol. 8(3): pp. 305–320, 2002.

[Mit92] David Mitchell, Bart Selman, and Hector Levesque: Hard and easy distributions of SAT problems. In *Proceedings of the tenth national conference on Artificial intelligence*, vol. 92 of *AAAI'92*, pp. 459–465. AAAI Press, 1992.

[Mla95] Nenad Mladenović: A variable neighborhood algorithm-a new metaheuristic for combinatorial optimization. In *papers presented at Optimization Days*, p. 112. 1995.

[Mla97] Nenad Mladenović and Pierre Hansen: Variable neighborhood search. *Computers & Operations Research*, vol. 24(11): pp. 1097–1100, 1997.

[Mok04] Ethel Mokotoff: An exact algorithm for the identical parallel machine scheduling problem. *European Journal of Operational Research*, vol. 152(3): pp. 758–769, 2004.

[Mon01] Douglas C. Montgomery: *Design and analysis of experiments*. John Wiley & Sons, 2001.

[Mor46] P. M. Morse and G. E. Kimball: Methods of operations research. OEG report 54, Office of the Chief of Naval Operations, US Navy Department, Washington, DC: US Superintendent of Documents, 1946.

[Mor12] Dominik Moritz and Matthias Springer: Solving Satisfiability with Ant Colony Optimization and Genetic Algorithms, 2012. Evaluation of project in Statistical optimization seminar at University of Potsdam.

[Mou11] Zohreh Mousavinasab, Reza Entezari-Maleki, and Ali Movaghar: A bee colony task scheduling algorithm in computational grids. In *Digital Information Processing and Communications*, vol. 188 of *Communications in Computer and Information Science*, pp. 200–210. Springer Berlin Heidelberg, 2011.

[MS08] Joao Marques-Silva: Practical applications of Boolean Satisfiability. In *Proceedings of the 9th International Workshop on Discrete Event Systems, Götebor, Sweden, May 28-30*, pp. 74–80. IEEE, 2008.

[Mum09] Christine L. Mumford: *Computational intelligence: collaboration, fusion and emergence*, vol. 1. Springer Science & Business Media, 2009.

[Nak03] Sunil Nakrani and Craig Tovey: On honey bees and dynamic allocation in an internet server colony. In *Proceedings of 2nd International Workshop on the Mathematics and Algorithms of Social Insects*. Citeseer, 2003.

[Nak13] Amir Nakib and Patrick Siarry: Performance analysis of dynamic optimization algorithms. In *Metaheuristics for dynamic optimization*, pp. 1–16. Springer, 2013.

[Nar09] Harikrishna Narasimhan: Parallel artificial bee colony (PABC) algorithm. In *World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009*, pp. 306–311. IEEE, 2009.

[Ned09] Ranko Nedeljkovic, Slobodan Mitrovic, and Dragana Drenovac: Bee colony optimization meta-heuristic for backup allocation problem. *Proc. PosTel XXVII*, pp. 115–122, 2009. In serbian.

[Nik10] Alexander G. Nikolaev and Sheldon H. Jacobson: Simulated annealing. In M. Gendreau and J-Y. Potvin (eds.), *Handbook of Metaheuristics*, pp. 1–39. (second edition) Springer, New York Dordrecht Heidelberg London, 2010.

[Nik13a] Miloš Nikolić and Dušan Teodorović: Empirical study of the Bee Colony Optimization (BCO) algorithm. *Expert Systems with Applications*, vol. 40(11): pp. 4609–4620, 2013.

[Nik13b] Miloš Nikolić and Dušan Teodorović: Transit network design by Bee Colony Optimization. *Expert Systems With Applications*, vol. 40(15): pp. 5945–5955, 2013.

[Nik15] Miloš Nikolić: *Disruption Management in Transportation by the Bee Colony Optimization Metaheuristic*. Ph.D. thesis, University of Belgrade, Faculty of Transport and Traffic Engineering, 2015. Adviser prof. Dr. Dušan Teodorović.

[Noc99] Jorge Nocedal and Stephen J. Wright: *Numerical optimization*. Springer-Verlag, USA, 1999.

[Ogn04] Zoran Ognjanović and Nenad Krdžavac: *Uvod u teorijsko računarstvo*. Beograd-Kragujevac, 2004.

[Osm96a] Ibrahim H. Osman and James P Kelly: *Meta-Heuristics: Theory and Applications*. Boston: Kluwe Academic Publisher, 1996.

[Osm96b] Ibrahim H. Osman and Gilbert Laporte: Metaheuristics: A bibliography. *Annals of Operations Research*, vol. 63(5): pp. 511–623, 1996.

[Pap91a] Christos H. Papadimitriou: On selecting a satisfying truth assignment. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pp. 163–169. IEEE, 1991.

[Pap91b] Christos H. Papadimitriou and Mihalis Yannakakis: Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, vol. 43(3): pp. 425–440, 1991.

[Pap91c] Athanasios Papoulis: *Probability, random variables, and stochastic processes*. McGraw-Hill, 1991.

[Pap98] Christos H. Papadimitriou and Kenneth Steiglitz: *Combinatorial optimization: algorithms and complexity*. Courier Dover Publication, New York, 1998.

[Pap07] Christos H. Papadimitriou: *Computational Complexity*. Cram101 Incorporated, 2007.

[Par11] Rafael Stubs Parpinelli, César Manuel Vargas Benitez, and Heitor Silvério Lopes: Parallel Approaches for the Artificial Bee Colony Algorithm. In B. K. Panigrahi, Y. Shi, and M-H. Lim (eds.), *Handbook of Swarm Intelligence*, pp. 329–345. Springer, 2011.

[Ped11] Martín Pedemonte, Sergio Nesmachnow, and Héctor Cancela: A survey on parallel ant colony optimization. *Applied Soft Computing*, vol. 11(8): pp. 5181–5197, 2011.

[Per11] Anggi Putri Pertiwi and Suyanto: Globally evolved dynamic bee colony optimization. In *Knowledge-Based and Intelligent Information and Engineering Systems*, pp. 52–61. Springer, 2011.

[Pha06] Duc Truong Pham, Afshin Ghanbarzadeh, Ebubekir Koç, Sameh Otri, S. Rahim, and Muhamad Zaidi: The bees algorithm–a novel tool for complex optimisation problems. In *Proceedings of the 2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS 2006)*, pp. 454–459. 2006.

[Pin84] J´nos Pintér: Convergence properties of stochastic optimization procedures. *Optimization*, vol. 15(3): pp. 405–427, 1984.

[Pin86] János Pintér: Contributions to the methodology of stochastic optimization. In *Stochastic programming*, pp. 247–257. Springer, 1986.

[Pin04] Michael L. Pinedo: Offline Deterministic Sscheduling, Stochastic Scheduling, and Online Deterministic Scheduling: A Comparative Overview. In Joseph Y-T. Leung (ed.), *Handbook of Scheduling, Chapman & Hall/CRC*. 2004. Chapter 38.

[Pin12] Michael L. Pinedo: *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2012.

[Pol07] Riccardo Poli, James Kennedy, and Tim Blackwell: Particle swarm optimization. *Swarm intelligence*, vol. 1(1): pp. 33–57, 2007.

[Pre04] Steven David Prestwich and Colin Quirke: Local Search for Very Large SAT Problems. In *Proceedings of the SAT 2004*, pp. 317–322. 2004.

[Que94] Maurice Queyranne and Andreas S. Schulz: Polyhedral approaches to machine scheduling. Tech. rep., Technische Universitat Berlin, 1994.

[Rah02] Malek Rahoual, Riad Hadji, and Vincent Bachelet: Parallel ant system for the set covering problem. In *Ant Algorithms*, pp. 262–267. Springer, 2002.

[Ran98] Soraya Rana and Darrell Whitley: Genetic Aalgorithm Behavior in the MAXSAT Domain. In *International Conference on Parallel Problem Solving from Nature*, pp. 785–794. Springer, 1998.

[Rar01] Ronald L. Rardin and Reha Uzsoy: Experimental evaluation of heuristic optimization algorithms: A tutorial. *Journal of Heuristics*, vol. 7(3): pp. 261–304, 2001.

[Ras63] Leonard Andreevich Rastrigin: About Convergence of Random Search Method in Extremal Control of Multi-Parameter System. *Automation and Remote Control*, vol. 24(11): pp. 1467–1473, 1963. Http://mi.mathnet.ru/at12312.

[Ras11] Reza Rastegar: On the optimal convergence probability of univariate estimation of distribution algorithms. *Evolutionary computation*, vol. 19(2): pp. 225–248, 2011.

[Ray12] Michel Raynal: *Concurrent programming: algorithms, principles, and foundations*. Springer Science & Business Media, 2012.

[Ree93] Colin R. Reeves: *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., 1993.

[Ree10] Colin R. Reeves: Genetic algorithms. In *Handbook of Metaheuristics*, pp. 109–139. Springer, 2010.

[Res10] Mauricio G.C. Resende and Celso C. Ribeiro: Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In *Handbook of Metaheuristics*, pp. 283–319. Springer, 2010.

[Ric06] John Rice: *Mathematical statistics and data analysis*. Thomson Brooks/Cole, 2006.

[Rid07] Enda Ridge: *Design of experiments for the tuning of optimisation algorithms*. University of York, Department of Computer Science, 2007.

[Rob09] Yves Robert and Frédéric Vivien: *Introduction to Scheduling*. CRC, Taylor & Francis Group, 2009.

[Roy82] J. P. Royston: An extension of Shapiro and Wilk's W test for normality to large samples. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, pp. 115–124, 1982.

[Rud94] Günter Rudolph: Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, vol. 5(1): pp. 96–101, 1994.

[Rud96] Günter Rudolph: Convergence of evolutionary algorithms in general search spaces. In *In Proceedings of the Third IEEE Conference on Evolutionary Computation*. 1996.

[Rud12] Günter Rudolph: Stochastic convergence. In *Handbook of Natural Computing*, pp. 847–869. Springer, 2012.

[Sa'13] Majid Sa'idi, Navid Mostoufi, and Rahmat Sotudeh-Gharebagh: Modelling and optimisation of continuous catalytic regeneration process using bee colony algorithm. *The Canadian Journal of Chemical Engineering*, vol. 91(7): pp. 1256–1269, 2013.

[Sam98] Diana Sammataro and Alphonse Avitabile: *The beekeeper's handbook*. Cornell University Press, 1998.

[Sat97] Takao Sato and Masafumih Hagiwara: Bee system: finding solution by a concentrated search. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 4, pp. 3954–3959. IEEE, 1997.

[Sch98] Alexander Schrijver: *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[Sch99] Uwe Schöning: A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pp. 410–414. IEEE, 1999.

[Sch01] Lothar M Schmitt: Theory of genetic algorithms. *Theoretical Computer Science*, vol. 259(1): pp. 1–61, 2001.

[See85] Thomas D. Seeley *et al.*: *Honeybee ecology: a study of adaptation in social life.* Princeton University Press, 1985.

[See91] Thomas D. Seeley, Scott Camazine, and James Sneyd: Collective decision-making in honey bees: how colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, vol. 28(4): pp. 277–290, 1991.

[See95] Thomas D. Seely: The wisdom of the hive. *Harvard UniversityPress, Cambridge, Massachusetts, USA*, 1995.

[See00] Thomas D. Seeley, Alexander S. Mikheyev, and Gary J. Pagano: Dancing bees tune both duration and rate of waggle-run production in relation to nectar-source profitability. *Journal of Comparative Physiology A*, vol. 186(9): pp. 813–819, 2000.

[Sel92] Bart Selman, Hector J. Levesque, and David G. Mitchell: A New Method for Solving Hard Satisfiability Problems. In *Proceedings of the tenth national conference on Artificial intelligence*, vol. 92 of *AAAI'92*, pp. 440–446. AAAI Press, 1992.

[Sel94] Bart Selman, Henry A Kautz, and Bram Cohen: Noise strategies for improving local search. In *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence*, AAAI'94, pp. 337–343. AAAI Press, 1994.

[Sel95] Bart Selman, Henry Kautz, Bram Cohen *et al.*: Local search strategies for satisfiability testing. *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge*, vol. 26: pp. 521–532, 1995.

[Sel96] Bart Selman, Henry Kautz, and Bram Cohen: Local Search Strategies for Satisfiability Testing. In *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*, pp. 521–532. American Mathematical Society, 1996.

[Šel08] Milica Šelmić, Praveen Edara, and Dušan Teodorović: Bee colony optimization approach to optimize locations of traffic sensors on highways. *Tehnika*, vol. 6: pp. 9–15, 2008.

[Šel10] Milica Šelmić, Dušan Teodorović, and Katarina Vukadinović: Locating inspection facilities in traffic networks: an artificial intelligence approach. *Transportation Planning and Technology*, vol. 33(6): pp. 481–493, 2010.

[Sev07] M. Sevkli and M.E. Aydin: Parallel variable neighbourhood search algorithms for job shop scheduling problems. *IMA Journal of Management Mathematics*, vol. 18(2): pp. 117–133, 2007.

[Sha96] Alexander Shapiro and Y. Wardi: Convergence analysis of stochastic algorithms. *Mathematics of Operations Research*, vol. 21(3): pp. 615–628, 1996.

[She04] David J. Sheskin: *Handbook of parametric and nonparametric statistical procedures*. Chapman & Hall/CRC, 2004.

[She11] David Sheldon: *Design Space Exploration of Parameterized Systems using Design of Experiments*. Ph.D. thesis, University of California, Riverside, 2011. Adviser prof. Dr. Frank Vahid.

[Sin07] Oliver Sinnen: *Task scheduling for parallel systems*, vol. 60. John Wiley & Sons, 2007.

[Smi12] Selmar Kagiso Smit: *Parameter tuning and scientific testing in evolutionary algorithms*. Ph.D. thesis, Vrije Universiteit, 2012.

[Smu12] Szesław Smutnicki: Optimization Technologies for Hard Problems. In János Fodor, Ryszard Klempous, and Carmen Paz Suárez Araujo (eds.), *Recent Advances in Intelligent Engineering Systems*, vol. 378 of *Studies in Computational Intelligence*, pp. 79–104. Springer, 2012.

[Soh11] Mohammad Falahi Sohi, Morteza Shirdel, and Ali Javidaneh: Applying BCO algorithm to solve the optimal DG placement and sizing problem. In *Power Engineering and Optimization Conference (PEOCO), 2011 5th International*, pp. 71–76. IEEE, 2011.

[Sol81] Francisco J. Solis and Roger J.-B. Wets: Minimization by random search techniques. *Mathematics of operations research*, vol. 6(1): pp. 19–30, 1981.

[Sör13] Kenneth Sörensen: Metaheuristics - the metaphor exposed. *International Transactions in Operational Research*, vol. 22(1): pp. 3–18, 2013.

[Spa03] James C. Spall: *Introduction to stochastic search and optimization: estimation, simulation, and control*. John Wiley & Sons, Hoboken, New Jersey, 2003.

[Sta07] Zorica Stanimirović: *Genetski algoritmi za rešavanje nekih NP-teških hab lokacijskih problema*. Ph.D. thesis, Matematički fakultet, Univerzitet u Beogradu, 2007.

[Ste46] Stanley S. Stevens: On the theory of scales of measurement. *Science*, vol. 103(2684): pp. 677–680, 1946.

[Ste00] Kathleen Steinhöfel, Andreas Albrecht, and Chak-Kuen Wong: Convergence analysis of simulated annealing-based algorithms solving flow shop scheduling problems. In *Algorithms and Complexity*, pp. 277–290. Springer, 2000.

[Ste02] Kathleen Steinhöfel, A. Albrecht, and Chak-Kuen Wong: Fast parallel heuristics for the job shop scheduling problem. *Computers & Operations Research*, vol. 29(2): pp. 151–169, 2002.

[Sto13] Tatjana Stojanović, Tatjana Davidović, and Zoran Ognjanović: Bee-colony optimization for the satisfiability problem in probabilistic logic. In *Treca nacionalna konferencija "Verovatnosne logike i njihove primene"*, p. 30. 2013.

[Sto15] Tatjana Stojanović, Tatjana Davidović, and Zoran Ognjanović: Bee colony optimization for the satisfiability problem in probabilistic logic. *Applied Soft Computing*, vol. 31: pp. 339–347, 2015.

[Stü98] Thomas Stützle: Parallelization strategies for ant colony optimization. In *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature*, vol. 1498, pp. 722–731. Springer, 1998.

[Stü01] Thomas Stützle, Holger Hoos, and Andrea Roli: A review of the literature on local search algorithms for MAX-SAT. Tech. Rep. AIDA-01-02, Intellectics Group, Darmstadt University of Technology, Germany, 2001.

[Stü02] Thomas Stützle and Marco Dorigo: A short convergence proof for a class of ant colony optimization algorithms. *IEEE Trans. Evolutionary Computation*, vol. 6(4): pp. 358–365, 2002.

[Stü09] Thomas Stützle, Mauro Birattari, and Holger H. Hoos (eds.): *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics, International Workshop, SLS 2009*, vol. 5752. Springer, September 2009. Preface.

[Sub11] Miloš Subotić, Milan Tuba, and Nadežda Stanarević: Different approaches in parallelization of the artificial bee colony algorithm. *International Journal of mathematical models and methods in applied sciences*, vol. 5(4): pp. 755–762, 2011.

[Sul01] Kelly Ann Sullivan and Sheldon H. Jacobson: A convergence analysis of generalized hill climbing algorithms. *IEEE Transactions on Automatic Control*, vol. 46(8): pp. 1288–1293, 2001.

[Tal09] El-Ghazali Talbi: *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.

[Teo01] Jason Teo and Hussein A. Abbass: An annealing approach to the mating-flight trajectories in the marriage in honey bees optimization algorithm. Tech. rep., School of computer Science, University of New South Wales of ADFA, 2001.

[Teo05] Dušan Teodorović and Mauro Dell'Orco: Bee colony optimization–a cooperative learning approach to complex transportation problems. In *Advanced OR and AI Methods in Transportation: Proceedings of 16th Mini–EURO Conference and 10th Meeting of EWGT (13-16 September 2005).–Poznan: Publishing House of the Polish Operational and System Research*, pp. 51–60. 2005.

[Teo06] Dušan Teodorović, Panta Lučić, Goran Marković, and Mauro Dell'Orco: Bee colony optimization: principles and applications. In *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*, pp. 151–156. IEEE, 2006.

[Teo07] Dušan Teodorović and Milica Šelmić: The BCO algorithm for the p-median problem. In *Proceedings of the XXXIV Serbian Operations Research Conferece*. Zlatibor, Serbia (in Serbian), 2007.

[Teo08] Dušan Teodorović and Mauro Dell'Orco: Mitigating traffic congestion: solving the ride-matching problem by bee colony optimization. *Transportation Planning and Technology*, vol. 31(2): pp. 135–152, 2008.

[Teo09a] Dušan Teodorović: Bee Colony Optimization (BCO). In *Innovations in Swarm Intelligence*, vol. 248 of *Studies in Computational Intelligence*, pp. 39–60. Springer Berlin Heidelberg, 2009.

[Teo09b] Dušan Teodorović: Bee colony optimization (BCO). In Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri (eds.), *Innovations in swarm intelligence*, vol. 248 of *Studies in Computational Intelligence*, pp. 39–60. Springer, 2009.

[Teo13] Dušan Teodorović, Milica Šelmić, and Ljiljana Mijatović-Teodorović: Combining case-based reasoning with Bee Colony Optimization for dose planning in well differentiated thyroid cancer treatment. *Expert Systems with Applications*, vol. 40(6): pp. 2147–2155, 2013.

[Teo15] Dušan Teodorović, Milica Šelmić, and Tatjana Davidović: Bee Colony Optimization Part II: The Application Survey. *Yugoslav Journal of Operational Research*, vol. 25(2): pp. 185–2019, 2015.

[The98] Arne Thesen: Design and evaluation of tabu search algorithms for multiprocessor scheduling. *Journal of Heuristics*, vol. 4(2): pp. 141–160, 1998.

[Tod13] Nikola Todorović and Sanja Petrović: Bee colony optimization algorithm for nurse rostering. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, vol. 43(2): pp. 467–473, 2013.

[Tre03] Ioan Cristian Trelea: The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, vol. 85(6): pp. 317–325, 2003.

[Tse73] Mikhail L'vovich Tsetlin: *Automaton theory and modeling of biological systems*, vol. 102. Academic Press New York, 1973.

[Tuk77] John W. Tukey: *Exploratory data analysis*. Addison-Wesley,Reading, Ma, 1977.

[Tur36] Alan Turing: On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, vol. 42: pp. 230–265, 1936.

[VDB06] Frans Van Den Bergh: *An analysis of particle swarm optimizers*. Ph.D. thesis, University of Pretoria, 2006.

[Ver95] Marcus Gerardus Aldegonda Verhoeven and Emile HL Aarts: Parallel local search. *Journal of Heuristics*, vol. 1(1): pp. 43–65, 1995.

[VF74] Karl Von Frisch: Decoding the language of the bee. *Science*, vol. 185(4152): pp. 663–668, 1974.

[Vil07] Marcos Villagra and Benjamín Barán: Ant Colony Optimization with Adaptive Fitness Ffunction for Satisfiability Ttesting. In *International Workshop on Logic, Language, Information, and Computation*, pp. 352–361. Springer, 2007.

[Voß12] Stefan Voß, Silvano Martello, Ibrahim H Osman, and Catherine Roucairol: *Meta-heuristics: Advances and trends in local search paradigms for optimization*. Springer Science & Business Media, 2012.

[Š11] Milica Šelmić: *Lociranje objekata na transportnim mrežama primenom metoda računarske inteligencije*. Ph.D. thesis, Saobraćajni fakultet, Univerzitet u Beogradu, 2011.

[Wan09] Xiaolei Wang: *Hybrid nature-inspired computation methods for optimization*. Ph.D. thesis, Helsinki University of Technology, Faculty of Eletronics, Communications and Automation, 2009.

[Wat10] Jean-Paul Watson: An introduction to fitness landscape analysis and cost models for local search. In *Handbook of metaheuristics*, pp. 599–623. Springer, 2010.

[Wed04] Horst F. Wedde, Muddassar Farooq, and Yue Zhang: BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In *Ant colony optimization and swarm intelligence*, pp. 83–94. Springer, 2004.

[Wes95] Stephen G. West, John F. Finch, and Patrick J. Curran: Structural equation models with nonnormal variables. *Structural equation modeling: Concepts, issues, and applications*, pp. 56–75, 1995.

[Whi70] R. C. White: A survey of random methods for parameter optimization. Tech. rep., 1970. No. 70-E-16.

[Win62] Ben James Winer: *Statistical Principles in Experimental Design*, vol. 1. McGraw-Hill, 1962.

[Won09] Li-Pei Wong, Malcolm Yoke Hean Low, and Chin Soon Chong: An efficient bee colony optimization algorithm for traveling salesman problem using frequency-based pruning. In *Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on*, pp. 775–782. IEEE, 2009.

[Won10a] Li-Pei Wong, Malcolm Yoke Hean Low, and Chin Soon Chong: Bee colony optimization with local search for traveling salesman problem. *International Journal on Artificial Intelligence Tools*, vol. 19(03): pp. 305–334, 2010.

[Won10b] Li-Pei Wong, Chi Yung Puan, Malcolm Yoke Hean Low, and Yi Wen Wong: Bee colony optimisation algorithm with big valley landscape exploitation for job shop scheduling problems. *International Journal of Bio-Inspired Computation*, vol. 2(2): pp. 85–99, 2010.

[Yan05] Xin-She Yang: Engineering Optimizations via Nature-inspired Virtual Bee Algorithms. In *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, pp. 317–323. Springer, 2005.

[Yan07a] Chenguang Yang, Jie Chen, and Xuyan Tu: Algorithm of fast marriage in honey bees optimization and convergence analysis. In *2007 IEEE International Conference on Automation and Logistics*, pp. 1794–1799. IEEE, 2007.

[Yan07b] Xin-She Yang: *A first course in finite element analysis*. Luniver Press, 2007.

[Yan07c] Zhongzhen Yang, Bin Yu, and Chuntian Cheng: A parallel ant colony algorithm for bus network optimization. *Computer-Aided Civil and Infrastructure Engineering*, vol. 22(1): pp. 44–55, 2007.

[Yan10] Xin-She Yang: *Engineering optimization: An Introduction with Metaheuristic Applications*. John Wiley & Sons, 2010.

[Yan11] Xin-She Yang: Metaheuristic optimization: algorithm analysis and open problems. In *Experimental Algorithms*, pp. 21–32. Springer, 2011.

[Yan13] Xin-She Yang, Suash Deb, Martin Loomes, and Mehmet Karamanoglu: A framework for self-tuning optimization algorithm. *Neural Computing and Applications*, vol. 23(7-8): pp. 2051–2057, 2013.

[Yan14] Xin-She Yang: Swarm intelligence based algorithms: a critical analysis. *Evolutionary Intelligence*, vol. 7(1): pp. 17–28, 2014.

[Yao99] Xin Yao: *Evolutionary computation: Theory and applications*. World Scientific, 1999.

[Yu11] Bin Yu, Z-Z Yang, and J-X Xie: A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society*, vol. 62(1): pp. 183–188, 2011.

[Zen04] Jian-Chao Zeng and Zhi-Hua Cui: A Guaranteed Global Convergence Particle Swarm Optimizer [J]. *Journal of computer research and development*, vol. 8: pp. 1333–1338, 2004.

[Zlo02] Mark Zlochin and Marco Dorigo: Model-based search for combinatorial optimization: A comparative study. In *Parallel Problem Solving from Nature—PPSN VII*, pp. 651–661. Springer, 2002.

[Zlo04] Mark Zlochin, Mauro Birattari, Nicolas Meuleau, and Marco Dorigo: Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, vol. 131(1-4): pp. 373–395, 2004.

APPENDIX A

Complementary material

A.1 Empirical study of BCOc

A.1.1 Time measurement

For measure of time *getrusage* function was used as it represents mot reliable way to measure CPU time on UNIX-style operating systems (http://nadeausoftware.com/articles/2012/03/c_c_tip_how_measure_cpu_time_benchmarking#times).

A.1.2 Box-plots

Box-plots are employed to graphically represent "five number summary" of the data in order to identify the minimum value, first quartile Q_1 (25%), median, third quartile Q_3 (75%) and the maximum. To generate box-plots, the data needs to be sorted in increasing order. The lowest value of the box-plot indicates the first value of this sorted list, which is also a minimum value of the complete dataset. It is straightforward to determine first 25% elements of the sorted list, the largest among them specifying the first quartile. In other words, first 25% of the sorted data resides in the range between the minimum and the first quartile. The median symbolizes the second quartile, i.e., the upper value of first 50% of the sorted data. The third quartile is determined by extracting elements of the sorted task list that are in the first 75%, and the value with highest index element represents the third quartile.

The box within box-plot represents the range between the first and the third quartile. In Figure 8.1 it indicates the range of processing times for at least 50% of all tasks of a particular instance. This value is also known in descriptive statistics as IQR (inner quartile range). For skewed data it is recommended to define *outliers*, i.e., data that is further away (far out) from a centralized grouping of the tasks lengths. The outliers are marked as red crosses and are positioned outside the interval $[Q_1 - 1.5\text{IQR}, Q_3 + 1.5\text{IQR}]$. The heights of the whiskers on box-plots indicate the first and the last data that are not outliers.

A.1.3 Candidate heuristics for $P||C_{max}$

In this section we provide description of four heuristic procedures that deal with the $P||C_{max}$ problem, modified from original versions found in the literature.

A.1.3.1 Solution representation

Following the conventions from [Dav12], the solution is represented as a (dynamic) matrix $S_{m\times n}$, where the element s_{ji} represents the index of the i-th task scheduled to the j-th machine. The list of all task lengths (processing time of each task) is kept in the vector $L = (l_1, \ldots, l_n)$. After allocating $k < n$ tasks, the auxiliary list of lengths of $n - k$ non-scheduled tasks is saved in the vector L'. Concerning the machine loads, two vectors are used: $O = (o_1, \ldots, o_m)$, where element o_j denotes the number of tasks allocated to j-th machine and $Y = (y_1, \ldots, y_m)$ with elements y_j that represent the sum of processing times of all tasks allocated to machine j.

At the end of iteration, data structure (S, O, Y, y_{max}) contains the obtained solution. Variable y_{max} indicates makespan of the solution. This solution is then compared with the the best-so-far (x_{best}), saved within a global structure g_{best} consisting of $(S_{gmin}, O_{gmin}, Y_{gmin}, y_{gmin})$, where y_{gmin} refers to its objective value.

In the forthcoming subsections, we give detailed description of four heuristics introduced previously. Common characteristic of all heuristics is the initialization phase, where the incorporated structure for solution representation is reinitialized at the beginning of each iteration (arrays S, O, Y contain all zeros, while $L' = L$).

A.1.3.2 Description of `sLPT+BF` heuristic

The heuristic algorithm `sLPT+BF` was inspired by well known *best-fit* algorithm, originally used for dealing with BPP. Best-fit algorithm dates back to the early 70s when it was applied to one-dimensional packing problems [Joh74]. Before implementing modified best-fit algorithm for the scheduling problem, it was necessary to define capacity of the machine loads (C). The lower bound on the *makespan* of tasks was used to initialize C, defined as the average of the sum of all tasks divided by the total number of machines. The *makespan* of the best found solution, reported at the competition of the iteration, is saved in the global variable y_{gmin}, while variable y contains running loads of all machines. In the first step of `sLPT+BF` the task i is chosen by stochastic LPT rule. The second step of `sLPT+BF` allocates task i onto the machine on which it leaves the least empty space, or in other words, is searching for the machine j with the smallest *gap* between the capacity C and the machine load after allocation of the task i. This is equivalent to searching for the biggest *makespan* of the partial solution after the allocation of task i. Another important feature of this algorithm is that C is changing every time best-so-far solution is found, i.e., C is set to be equal to the *makespan* (y_{gmin}) of the corresponding solution. This algorithm was used in the paper of Davidović et al. [Dav12]. Pseudo-code for `sLPT+BF` is provided in Fig. A.1. In each iteration the algorithm starts with an empty solution and constructs complete solutions. To select task-machine pairs the algorithm uses `sLPT+BF` rule. The iterations are repeated until a given stopping criterion is satisfied.

Some other scenarios for setting values of C were also studied and compared. For example, if in the initialization phase the variable C wasn't assigned a lower bound of the tasks lengths and was, instead, assigned a value larger than the lower bound, the algorithm generates solutions of lower quality. Moreover, if C is kept constant throughout the execution, the quality of the reported solution is not reaching the quality of the recommended procedure presented in Fig. A.1. The experimental tests, conducted on different instances of the scheduling problem, indicate that the variable C needs to be

Initialization: Read problem data and stopping criterion.
$C \leftarrow (\sum_{i=1}^{n} l_i)/m$;
$y_{gmin} =$ `MAX_INT`;
Do
  Assign an empty solution;
  for $(i = 0; i < n; i{+}{+})$
    (1) Choosing task i using the roulette wheel of size $\sum_{i=1}^{k} l_i$;
    (2) *if* $(i == 0)$
      Allocate first task i to random machine jj;
    else
      $min_{gap} \leftarrow$ `MAX_INT`;
      for $(j = 0; j < m; j{+}{+})$
        $gap \leftarrow C - (y_j + l_i)$;
        if ($gap > 0$ && $gap < min_{gap}$)
          $jj \leftarrow j$;
          $min_{gap} \leftarrow gap$;
      Allocate task i to machine jj;
  (3) // Find *makespan*
  $y_{max} \leftarrow 0$;
  for $(j = 0; j < m; j{+}{+})$;
    if $(y_{max} < y_j)$
      $y_{max} \leftarrow y_j$;
  (4) If *best-so-far* solution is found update necessary data:
  if $(y_{max} < y_{gmin})$;
    $y_{gmin} \leftarrow y_{max}$
    $C \leftarrow y_{gmin}$;
    Update g_{best};
while stopping criterion is not satisfied.
return $(x_{best}, f(x_{best}))$.

Figure A.1: Pseudo-code for `sLPT+BF`

set as suggested in Fig. A.1.

A.1.3.3 Description of `sLPT+FF` heuristic

Heuristic `sLPT+FF` represents a randomized iterative procedure that in each iteration starts with an empty solution, and similarly to the `sLPT+BF` algorithm, constructs complete solutions (Fig.A.2).

During the construction of the solution the algorithm goes through two steps, each incorporating some decision-making rules concerning the partial solution. In the first step, the decision about which tasks are to be scheduled is made by the rules of `sLPT`. Second step consists of decisions referring to the location of a task, by employing *first-fit* scheduler as suggested in [Joh74]. The machine selection procedure is implemented similarly to the second step of `sLPT+BF`, i.e., as a loop that visits all machines, however, placing the task onto the first machine on which it can be allocated. At the completion

of each iteration, all global structures are updated. The best solution is reported at the end of the execution of the algorithm.

Initialization: Read problem data and stopping criterion.
$C \leftarrow (\sum_{i=1}^{n} l_i)/m$;
$y_{gmin} =$ `MAX_INT`;
Do
    Assign an empty solution;
    for $(i = 0; i < n; i++)$
        (1) Choosing task i using the roulette wheel of size $\sum_{i=1}^{k} l_i$;
        (2) *if* $(i == 0)$
            Allocate first task i to random machine jj;
        else
            for $(j = 0; j < m; j++)$
                $gap \leftarrow C - (y_j + l_i)$;
                if ($gap > 0$)
                $jj \leftarrow j$;
            Allocate task i to machine jj;
    (3) Find current *makespan* y_{max}
    (4) If *best-so-far* solution is found update necessary data:
    if $(y_{max} < y_{gmin})$;
        $y_{gmin} \leftarrow y_{max}$
        $C \leftarrow y_{gmin}$;
        Update g_{best};
while stopping criterion is not satisfied.
return $(x_{best}, f(x_{best}))$.

Figure A.2: Pseudo-code for `sLPT+FF`

A.1.3.4 Description of `sLPT+ES`

`sLPT+ES` heuristic is a simple algorithm that begins with an empty solution and until the end of one iteration constructs the solution following two steps. First step is based on the stochastic procedure inspired by the `sLPT` rule, where the probability of choosing a task is proportional to its processing time. The selection of tasks was implemented using concepts of roulette wheel (Section 4.2, pg. 72). Second step incorporates simple earliest start scheduler. It is implemented as a loop that goes through all machines, and once the machine with minimum load is found, second phase ends. At the end of each iteration, global variable y_{gmin} is updated. Pseudo-code of the algorithm is provided in Fig. A.3.

A.1.3.5 Description of `sLPT+sES` algorithm

Another algorithm is studied in order to examine more variations of the scheduling rules. This algorithm consists of two steps, stochastic LPT and the stochastic ES. Pseudo-code is provided in Fig. A.4. The step (2) implements `sES` such that the machines with smallest load have higher probabilities to be chosen. This was conducted

Initialization: Read problem data and stopping criterion.
$y_{gmin} =$ `MAX_INT`;
Do
  Assign an empty solution;
  for $(i = 0; i < n; i++)$
    (1) Choosing task i by roulette wheel of size $\sum_{i=1}^{k} l_i$;
    $min \leftarrow$ `MAX_INT`;
    (2) *if* $(i == 0)$
      Allocate first task i to random machine jj;
    else
      if $(min > y_j)$
        $min = y_j$;
        $jj = j$;
      Allocate task i to machine jj;
  (3) Find current *makespan* y_{max}
  (4) If *best-so-far* solution is found update necessary data:
  if $(y_{max} < y_{gmin})$;
    $y_{gmin} \leftarrow y_{max}$
    Update g_{best};
while stopping criterion is not satisfied.
return $(x_{best}, f(x_{best}))$

Figure A.3: Pseudo-code for `sLPT+ES`

by using roulette wheel with wedges being proportional to the difference between a predefined maximal load and a current load of a machine. For that reason, capacity C is used (introduced in `sLPT+BF`) and updated each time new best-so-far solution is found.

A.1.4 Comparative study of candidate heuristics

To single out best performing heuristic procedure for the considered scheduling problem, we compare four candidate algorithms presented above. To assure systematic experimental evaluation for the comparison between different algorithms, some of the rules from the literature have been followed. First of all, the tests are based on comparing average performance of algorithm instances over a class of problem instances [Dav06b, Dav06a].

The solution's quality of the results generated by the BCO algorithm, corresponds to the best found makespan value (y_{gmin}) for each problem instance. We employ number of iterations (N_{it}) as a stopping criterion.

A.1.4.1 Discussion for variable C

As mentioned in Section 8.1.3, the first implementation of BCO for the scheduling problem was presented in [Dav09]. Authors based heuristic rules on the stochastic version of the LPT procedure, and it showed promising results. Better solutions were obtained

Initialization: Read problem data and stopping criterion.
$C \leftarrow (\sum_{i=1}^{n} l_i)/m$;
$y_{gmin} =$ `MAX_INT`;
Do
    Assign an empty solution;
    for $(i = 0; i < n; i++)$
        (1) Choosing task i using the roulette wheel of size $\sum_{i=1}^{k} l_i$;
        (2) Scheduling i on machine j using the roulette wheel
            of size $\sum_{j=1}^{m}(C - y_j)$;
    (3) Find current *makespan* y_{max}
    (4) If *best-so-far* solution is found update necessary data:
    if $(y_{max} < y_{gmin})$;
        $y_{gmin} \leftarrow y_{max}$
        $C \leftarrow y_{gmin}$;
        Update g_{best};
while stopping criterion is not satisfied.
return $(x_{best}, f(x_{best}))$

Figure A.4: Pseudo-code for `sLPT+sES` algorithm.

in [Dav12], where authors incorporated best-fit heuristic for selection of processors. They used adaptive strategy in order to change values of capacity C and direct the search towards promising areas of the search space. Prior to implementation of best-fit technique, we tested several possibilities of setting the value of C in order to investigate its influence on the performance of the heuristic and its robustness with respect to changes in the problem specifications. The pilot studies concerning the influence of parameter C on the constructive moves during the execution show that it should not be set as a constant. If C is close to known optimum, low quality solutions will be generated since tasks that don't pass the scheduling criterion will be allocated to the last visited machine. Next, we appointed C with values larger than the theoretical lower bound of the processor loads at the beginning of the search. This generated less variability in response values, however, degraded the quality of solution. Results indicate the `sLPT+BF` heuristic requries well defined threshold (capacity) in order to produce high quality results. Parameter C cannot be set provisory as the minimum capacity corresponds to the makespan of the optimal solution. Straightforward way is to define C in each iteration as the makespan of the best-so-far solution. Changing C gradually might direct the search towards more prominent areas of the search space. This way we can favor the generation of solutions that are not worse than the previous one. Our proposal is that in first iteration parameter C is set to the theoretical lower bound. Then, in the next iteration C is set to makespan of the best-so-far solution in order to allow generation of solutions that might be worse then the current. Presented mixed procedure has provided the best results. Therefore, we confirm that strategy of [Dav12] provides best results.

In [Dav12] it was concluded that n does not influence the complexity of the problem. The studies in this thesis confirm the same conclusion. We show that the influence of the varying number of processors on the complexity of the problem is not so straight-

forward. This problem is referred to in greater detail in the forthcoming section.

A.1.5 Experimental evaluation of the best heuristic: `sLPT+BF`

In the previous part of this section we demonstrated that `sLPT+BF` dominates other candidate heuristics on the hardest test instance from [Dav12]. Experimental evaluation of `sLPT+BF` on benchmark instances of [Dav06b, Dav06a] was not reported previously in the literature. Therefore, to understand performance of BCO, it was justifiable to conduct independent analysis of `sLPT+BF` for the scheduling problem. In the following, we report results of the experimental study of the `sLPT+BF`. We address several objectives while investigation performance of a standalone `sLPT+BF`. Firstly, we investigate stagnation of the solution by increasing maximal number of iterations. Stagnation of the solution occurs when, after reaching a certain number of iterations, heuristic does not generate solutions of significantly better quality. This is a common property of stochastic optimization methods. Furthermore, it is our goal to avoid so-called *ceiling effect*, where two algorithms are not distinguishable when provided enough time or iterations to reach a sub-optimal state [Coh95, BB13]. Therefore, we use new results on `sLPT+BF` to investigate a suitable value of maximal number of iterations for BCO. Lastly, we investigate robustness of `sLPT+BF` to changes in instances characteristics, such as dimension of a problem (number of tasks).

To analyze robustness of `sLPT+BF` a set of independent experiments was conducted on all 18 test instances from two different problem classes, $m = 12$ and $m = 16$. Instances of a class differ in number of tasks that can vary from 100 to 500. The choice of instances was based on the results from previous subsection, where it was concluded that if the number of machines is less then 12 ($m < 12$), problems are being solved to near optimality. In order to observe the influence of number of iterations on the quality of a solution, parameter N_{it} was set to different values within interval $[100, 10000]$. Therefore, one experiment corresponds to $n_{run} = 100$ independent runs of `sLPT+BF` (each with different seed) for appointed number of iterations (N_{it}). The results of this set of experiments are presented in Tables A.1 and A.2. Tables are organized in six sections, grouped by value of the stopping criterion. Rows are arranged by the type of instance, i.e., the number of tasks to be scheduled. Each row section consist of two rows, where the first one reports the means (average values), while the second row holds standard deviations and the number of occurrences of the best found solution. In the column sections, the first column ($\overline{y}$) represents a mean value of the best found solutions' quality, averaged over n_{run} independent runs. Second column ($\bar{t}$) represents mean value of running times needed to obtain best found solutions, reported in miliseconds. Third column ($\overline{n_{it}}$) represents mean of the number of iterations needed to generate best found solution and the forth column (*best*) reports the best found solutions of an experiment. Additional to the reported *best*, in both tables we provide percentage of its occurrence during the experiment.

A.1.5.1 Results

The analysis of responses, reported in Tables A.1 and A.2 and measured for different values of stopping criterion, is suggesting that the `sLPT+BF` heuristic needs some minimum number of iterations to reach position in the search space that increases its

Table A.1: Scheduling results - test problems with known optimal solutions adopted from [Dav06b] with $m = 12, 16$, $n_{run} = 100$ and a different number of iterations ($N_{it} = 100, \ldots, 1000$).

Probl. inst.	N_{it}=100				N_{it}=200				N_{it}=300				N_{it}=400				N_{it}=500				N_{it}=1000			
	$\bar{y}$	$\bar{t}$	$\overline{n_{it}}$	best	$\bar{y}$	$\bar{t}$	$\overline{n_{it}}$	best	$\bar{y}$	$\bar{t}$	$\overline{n_{it}}$	best	$\bar{y}$	$\bar{t}$	$\overline{n_{it}}$	best	$\bar{y}$	$\bar{t}$	$\overline{n_{it}}$	best	$\bar{y}$	$\bar{t}$	$\overline{n_{it}}$	best
	(s.d.)	(s.d.)	(s.d.)	(%)	(s.d.)	(s.d.)	(s.d.)	(%)	(s.d.)	(s.d.)	(s.d.)	(%)	(s.d.)	(s.d.)	(s.d.)	(%)	(s.d.)	(s.d.)	(s.d.)	(%)	(s.d.)	(s.d.)	(s.d.)	(%)
		$[10^{-3}]$				$[10^{-3}]$				$[10^{-3}]$				$[10^{-3}]$				$[10^{-3}]$				$[10^{-3}]$		
Iogra100_12	831.21	1.11	94.56	**817**	819.16	2.24	161.41	**812**	817.60	3.32	220.48	**812**	816.73	4.25	283.01	**812**	816.14	5.35	353.25	**811**	814.61	8.04	605.81	**811**
	± 17.10	± 0.55	± 6.94	(1%)	± 2.00	± 0.79	± 26.76	(1%)	± 1.76	± 1.20	± 53.00	(1%)	± 1.68	± 1.61	± 81.78	(1%)	± 1.71	± 2.01	± 104.19	(1%)	± 1.46	± 3.37	± 223.59	(5%)
16	903.25	1.54	99.64	**816**	815.16	3.03	174.85	**811**	812.43	4.21	240.67	**810**	811.66	5.19	293.70	**809**	811.19	5.79	354.26	**809**	810.00	8.36	597.71	**807**
	± 62.26	± 0.62	± 1.69	(4%)	± 5.99	± 0.74	± 18.19	(5%)	± 1.34	± 1.05	± 41.91	(8%)	± 1.12	± 1.22	± 57.12	(4%)	± 1.10	± 2.08	± 85.33	(7%)	± 0.94	± 3.69	± 203.72	(1%)
Iogra150_12	1022.35	2.28	94.46	**1008**	1008.64	4.15	157.23	**1006**	1007.53	5.46	217.08	**1005**	1007.13	6.04	264.33	**1004**	1006.76	7.93	322.64	**1004**	1005.91	12.06	578.31	**1004**
	± 23.72	± 0.71	± 9.04	(2%)	± 1.12	± 1.19	± 32.34	(3%)	± 1.04	± 1.90	± 52.62	(2%)	± 0.98	± 2.28	± 72.06	(1%)	± 0.90	± 3.15	± 104.24	(1%)	± 0.78	± 4.96	± 231.48	(3%)
16	1106.23	2.83	99.60	**1014**	1015.81	5.50	180.05	**1011**	1011.89	7.13	238.52	**1008**	1010.98	8.12	300.51	**1008**	1010.57	9.29	346.36	**1008**	1009.35	15.34	649.32	**1007**
	± 65.90	± 0.65	± 1.57	(1%)	± 10.34	± 1.03	± 18.32	(3%)	± 1.17	± 1.63	± 39.84	(1%)	± 1.08	± 2.38	± 67.71	(2%)	± 1.09	± 3.15	± 85.22	(4%)	± 1.03	± 4.94	± 222.67	(1%)
Iogra200_12	1212.94	3.43	89.09	**1206**	1207.20	5.37	142.22	**1205**	1206.60	6.90	193.26	**1204**	1206.17	8.59	256.12	**1204**	1205.91	10.00	306.37	**1204**	1205.29	15.36	506.29	**1204**
	± 10.07	± 0.90	± 12.29	(3%)	± 0.92	± 1.68	± 34.76	(3%)	± 0.95	± 2.65	± 60.34	(3%)	± 0.86	± 3.63	± 91.71	(5%)	± 0.83	± 3.64	± 107.05	(6%)	± 0.67	± 6.68	± 224.53	(12%)
16	1335.63	4.45	99.74	**1215**	1226.18	8.05	185.23	**1210**	1211.61	10.40	251.74	**1209**	1210.74	10.98	302.08	**1208**	1210.17	13.09	371.36	**1208**	1208.80	22.85	676.39	**1207**
	± 77.85	± 0.73	± 1.38	(1%)	± 30.55	± 1.69	± 19.30	(1%)	± 1.30	± 2.42	± 36.30	(4%)	± 1.05	± 2.52	± 61.17	(3%)	± 0.97	± 3.33	± 89.13	(5%)	± 0.91	± 6.51	± 206.55	(8%)
Iogra250_12	1408.33	4.80	86.46	**1405**	1405.89	6.73	137.76	**1404**	1405.40	8.58	186.67	**1404**	1405.19	9.85	223.00	**1404**	1405.01	11.49	265.00	**1403**	1404.47	20.38	484.26	**1403**
	± 6.17	± 1.08	± 10.81	(3%)	± 0.75	± 2.19	± 33.78	(1%)	± 0.63	± 3.47	± 61.49	(5%)	± 0.59	± 3.99	± 82.77	(9%)	± 0.64	± 4.88	± 109.67	(1%)	± 0.64	± 9.55	± 235.74	(5%)
16	1499.30	6.26	99.16	**1410**	1416.86	10.34	182.83	**1407**	1408.66	12.25	242.92	**1406**	1407.83	14.69	301.42	**1406**	1407.41	17.34	360.57	**1405**	1406.66	26.69	583.92	**1405**
	± 62.22	± 0.97	± 3.83	(4%)	± 18.31	± 1.75	± 20.34	(1%)	± 1.11	± 2.16	± 42.73	(2%)	± 0.91	± 3.54	± 65.75	(9%)	± 0.79	± 4.07	± 86.95	(1%)	± 0.70	± 9.37	± 216.17	(7%)
Iogra300_12	1607.53	6.74	86.32	**1605**	1605.37	9.77	143.95	**1603**	1604.92	11.84	188.98	**1603**	1604.65	14.38	231.84	**1603**	1604.40	17.02	280.79	**1602**	1603.98	26.90	452.81	**1602**
	± 1.99	± 1.50	± 11.58	(5%)	± 0.77	± 3.08	± 37.08	(1%)	± 0.70	± 3.56	± 53.56	(2%)	± 0.67	± 5.26	± 82.97	(3%)	± 0.62	± 6.84	± 114.77	(1%)	± 0.65	± 14.18	± 248.74	(2%)
16	1664.35	8.71	99.27	**1609**	1609.24	12.51	170.99	**1606**	1607.75	15.52	218.19	**1606**	1607.29	18.63	272.14	**1606**	1606.97	22.58	332.69	**1605**	1606.44	34.44	524.69	**1605**
	± 42.56	± 1.60	± 2.48	(2%)	± 2.72	± 1.99	± 24.33	(3%)	± 0.80	± 3.16	± 43.60	(7%)	± 0.75	± 5.17	± 77.63	(13%)	± 0.67	± 6.42	± 98.55	(2%)	± 0.67	± 14.26	± 227.76	(7%)
Iogra350_12	1826.67	8.83	95.63	**1811**	1810.79	13.09	158.33	**1808**	1809.87	17.07	213.80	**1807**	1809.34	21.04	273.74	**1806**	1808.83	26.02	341.95	**1806**	1807.96	40.70	552.69	**1806**
	± 22.13	± 1.70	± 7.13	(7%)	± 1.25	± 2.52	± 26.51	(4%)	± 1.00	± 4.20	± 52.10	(2%)	± 1.07	± 5.86	± 78.97	(1%)	± 1.09	± 7.69	± 105.16	(2%)	± 0.89	± 15.29	± 216.89	(4%)
16	1850.14	10.42	99.14	**1809**	1808.19	14.80	163.60	**1805**	1807.31	18.34	215.68	**1805**	1806.89	22.50	265.21	**1805**	1806.68	25.35	303.11	**1805**	1806.14	40.38	495.94	**1804**
	± 33.85	± 1.70	± 2.83	(2%)	± 0.89	± 2.31	± 25.56	(1%)	± 0.78	± 4.19	± 48.08	(1%)	± 0.68	± 5.68	± 71.69	(2%)	± 0.69	± 7.78	± 97.84	(3%)	± 0.66	± 18.08	± 232.72	(1%)
Iogra400_12	2025.39	10.99	95.59	**2005**	2007.14	15.29	158.58	**2005**	2006.37	19.36	203.27	**2005**	2005.87	24.70	263.50	**2004**	2005.54	29.86	323.77	**2004**	2004.75	51.06	572.16	**2003**
	± 23.65	± 1.81	± 8.50	(1%)	± 0.98	± 2.80	± 27.89	(5%)	± 0.76	± 4.77	± 50.44	(12%)	± 0.69	± 7.34	± 84.26	(1%)	± 0.70	± 9.14	± 105.22	(5%)	± 0.73	± 18.80	± 218.31	(5%)
16	2085.98	12.91	99.49	**2010**	2011.56	18.81	177.10	**2006**	2008.19	23.99	235.45	**2006**	2007.67	29.02	283.54	**2006**	2007.20	35.26	351.08	**2006**	2006.37	58.56	594.30	**2005**
	± 50.56	± 2.06	± 2.50	(2%)	± 7.86	± 2.38	± 22.48	(1%)	± 0.89	± 4.24	± 42.88	(2%)	± 0.85	± 6.64	± 65.49	(7%)	± 0.68	± 8.71	± 90.65	(12%)	± 0.61	± 20.66	± 216.57	(7%)
Iogra450_12	2235.32	12.76	96.15	**2206**	2208.35	18.80	161.48	**2205**	2207.38	24.44	217.99	**2205**	2206.59	31.28	283.31	**2205**	2206.28	36.95	337.58	**2205**	2205.50	62.74	582.80	**2204**
	± 34.41	± 2.05	± 7.70	(1%)	± 1.24	± 3.32	± 28.25	(1%)	± 0.89	± 5.54	± 51.60	(2%)	± 0.85	± 8.07	± 78.08	(11%)	± 0.75	± 9.93	± 95.55	(12%)	± 0.69	± 23.31	± 224.54	(5%)
16	2349.58	13.58	99.69	**2212**	2239.97	23.53	191.14	**2209**	2211.84	31.26	258.29	**2208**	2210.18	37.76	316.26	**2208**	2209.50	44.70	377.78	**2207**	2208.22	78.43	670.94	**2206**
	± 79.44	± 1.26	± 1.47	(1%)	± 41.99	± 2.25	± 14.82	(3%)	± 3.00	± 4.40	± 37.96	(1%)	± 1.19	± 7.07	± 61.27	(7%)	± 0.94	± 9.74	± 84.63	(2%)	± 0.81	± 20.84	± 184.04	(1%)
Iogra500_12	2412.72	12.78	89.11	**2406**	2406.04	19.82	147.04	**2404**	2405.43	26.30	198.24	**2403**	2405.09	33.52	257.51	**2403**	2404.85	38.12	294.43	**2403**	2404.31	63.80	504.30	**2403**
	± 11.07	± 2.24	± 12.13	(7%)	± 0.92	± 4.29	± 33.07	(6%)	± 0.75	± 6.86	± 56.08	(1%)	± 0.66	± 9.99	± 80.43	(1%)	± 0.64	± 13.40	± 107.23	(2%)	± 0.66	± 29.35	± 240.83	(10%)
16	2515.92	15.59	99.71	**2414**	2416.10	26.70	186.12	**2409**	2409.92	34.57	244.89	**2407**	2409.21	41.95	300.99	**2407**	2408.77	49.73	359.18	**2406**	2407.86	82.42	603.14	**2406**
	± 52.60	± 1.27	± 2.59	(2%)	± 11.78	± 2.38	± 14.48	(2%)	± 0.96	± 5.12	± 36.30	(1%)	± 0.79	± 7.86	± 58.23	(1%)	± 0.76	± 11.14	± 82.92	(1%)	± 0.75	± 27.78	± 209.96	(4%)

Table A.2: Scheduling results - test problems with known optimal solutions adopted from [Dav06b] with $m = 12, 16$, $n_{run} = 100$ and different number of iterations ($N_{it} = 2000, \ldots, 10000$) (cont).

Probl. inst.	N_{it}=2000				N_{it}=4000				N_{it}=5000				N_{it}=6000				N_{it}=8000				N_{it}=10000			
	$\bar{y}$ (s.d.)	$\bar{t}$ (s.d.) [10^{-3}]	$\overline{n_{it}}$ (s.d.)	best (%)	$\bar{y}$ (s.d.)	$\bar{t}$ (s.d.) [10^{-3}]	$\overline{n_{it}}$ (s.d.)	best (%)	$\bar{y}$ (s.d.)	$\bar{t}$ (s.d.) [10^{-3}]	$\overline{n_{it}}$ (s.d.)	best (%)	$\bar{y}$ (s.d.)	$\bar{t}$ (s.d.) [10^{-3}]	$\overline{n_{it}}$ (s.d.)	best (%)	$\bar{y}$ (s.d.)	$\bar{t}$ (s.d.) [10^{-3}]	$\overline{n_{it}}$ (s.d.)	best (%)	$\bar{y}$ (s.d.)	$\bar{t}$ (s.d.) [10^{-3}]	$\overline{n_{it}}$ (s.d.)	best (%)
Iogra100_12	813.09	15.35	1222.34	**809**	812.01	27.46	2344.87	**808**	811.74	32.87	2799.97	**808**	811.50	37.59	3251.44	**808**	811.17	49.63	4352.36	**808**	810.96	58.58	5171.31	**808**
	± 1.27	± 5.70	± 474.27	(1%)	± 1.23	± 10.71	± 952.01	(1%)	± 1.14	± 13.66	± 1221.15	(1%)	± 1.10	± 16.87	± 1517.09	(1%)	± 1.05	± 23.00	± 2103.30	(1%)	± 1.01	± 28.72	± 2612.75	(1%)
16	809.26	14.67	1043.68	**807**	808.45	29.69	2228.49	**806**	808.17	36.56	2779.86	**805**	807.98	42.76	3256.97	**805**	807.77	52.98	4092.97	**805**	807.60	65.09	5083.61	**805**
	± 0.86	± 6.74	± 490.78	(4%)	± 0.84	± 12.77	± 1015.61	(1%)	± 0.87	± 16.36	± 1313.63	(1%)	± 0.86	± 19.21	± 1554.93	(1%)	± 0.81	± 25.50	± 2040.68	(1%)	± 0.72	± 31.08	± 2508.24	(1%)
Iogra150_12	1005.22	22.36	1094.62	**1004**	1004.74	36.12	1850.28	**1003**	1004.55	47.71	2470.96	**1003**	1004.42	53.68	2794.38	**1003**	1004.18	72.09	3797.58	**1003**	1004.04	87.45	4643.54	**1003**
	± 0.69	± 9.30	± 482.26	(12%)	± 0.61	± 18.61	± 991.57	(2%)	± 0.61	± 25.26	± 1376.09	(4%)	± 0.62	± 29.50	± 1606.20	(7%)	± 0.61	± 39.42	± 2146.98	(11%)	± 0.60	± 47.66	± 2602.35	(16%)
16	1008.59	26.10	1124.66	**1006**	1007.95	46.12	2086.50	**1006**	1007.67	57.89	2656.31	**1005**	1007.49	69.32	3198.37	**1005**	1007.23	85.64	3986.68	**1005**	1007.02	111.05	5190.11	**1005**
	± 0.84	± 9.36	± 447.46	(1%)	± 0.79	± 22.21	± 1058.68	(3%)	± 0.80	± 27.58	± 1320.45	(1%)	± 0.77	± 32.95	± 1584.25	(1%)	± 0.79	± 42.68	± 2038.43	(2%)	± 0.71	± 52.95	± 2525.87	(2%)
Iogra200_12	1204.81	28.42	953.68	**1204**	1204.34	53.36	1867.51	**1203**	1204.14	70.03	2461.02	**1203**	1204.02	80.81	2871.53	**1203**	1203.86	100.13	3572.32	**1203**	1203.79	113.01	4041.43	**1203**
	± 0.54	± 14.86	± 529.24	(26%)	± 0.51	± 29.92	± 1084.00	(1%)	± 0.49	± 36.67	± 1322.93	(6%)	± 0.45	± 41.64	± 1516.61	(9%)	± 0.51	± 56.39	± 2060.46	(21%)	± 0.50	± 66.01	± 2412.35	(25%)
16	1207.93	39.24	1189.79	**1206**	1207.23	68.21	2131.10	**1206**	1207.04	81.41	2553.25	**1206**	1206.92	94.51	2981.28	**1205**	1206.67	126.07	4010.52	**1205**	1206.54	145.93	4653.37	**1205**
	± 0.85	± 12.85	± 417.88	(4%)	± 0.68	± 30.71	± 994.29	(12%)	± 0.69	± 38.63	± 1234.25	(20%)	± 0.67	± 45.89	± 1490.64	(1%)	± 0.66	± 63.94	± 2073.81	(2%)	± 0.64	± 77.02	± 2496.41	(2%)
Iogra250_12	1404.10	34.68	840.79	**1403**	1403.64	71.79	1799.26	**1403**	1403.50	88.48	2239.78	**1403**	1403.46	95.75	2423.02	**1402**	1403.32	128.02	3260.74	**1402**	1403.25	149.11	3796.38	**1402**
	± 0.54	± 18.93	± 488.53	(10%)	± 0.54	± 42.66	± 1101.81	(39%)	± 0.50	± 50.71	± 1314.66	(50%)	± 0.52	± 56.25	± 1458.55	(1%)	± 0.49	± 78.03	± 2028.61	(1%)	± 0.46	± 95.75	± 2467.69	(1%)
16	1406.06	45.17	1007.26	**1404**	1405.66	75.28	1717.19	**1404**	1405.49	96.19	2209.54	**1404**	1405.33	121.15	2801.73	**1404**	1405.04	182.38	4217.18	**1404**	1404.89	216.12	4983.94	**1404**
	± 0.65	± 19.03	± 444.93	(1%)	± 0.65	± 41.34	± 971.42	(2%)	± 0.67	± 56.92	± 1332.61	(6%)	± 0.62	± 73.85	± 1733.98	(7%)	± 0.55	± 97.43	± 2282.39	(13%)	± 0.55	± 114.58	± 2657.18	(21%)
Iogra300_12	1603.60	50.41	860.12	**1602**	1603.31	84.49	1472.88	**1602**	1603.22	103.05	1805.70	**1602**	1603.07	141.88	2501.95	**1602**	1602.97	173.07	3041.29	**1602**	1602.93	188.66	3334.45	**1602**
	± 0.58	± 28.69	± 515.53	(3%)	± 0.59	± 59.01	± 1056.65	(7%)	± 0.58	± 74.27	± 1329.59	(8%)	± 0.50	± 99.88	± 1792.14	(9%)	± 0.43	± 120.80	± 2156.47	(11%)	± 0.41	± 134.58	± 2406.08	(12%)
16	1605.92	61.34	950.69	**1605**	1605.44	111.04	1749.94	**1604**	1605.35	131.66	2079.59	**1604**	1605.27	149.18	2361.02	**1604**	1605.08	215.82	3435.39	**1604**	1604.96	264.82	4222.53	**1604**
	± 0.63	± 29.52	± 477.47	(24%)	± 0.55	± 63.57	± 1021.67	(2%)	± 0.57	± 79.22	± 1274.48	(4%)	± 0.56	± 93.92	± 1507.85	(6%)	± 0.52	± 136.84	± 2204.03	(10%)	± 0.53	± 168.84	± 2721.23	(16%)
Iogra350_12	1807.06	81.75	1130.74	**1805**	1806.51	145.78	2038.92	**1805**	1806.36	169.61	2378.59	**1805**	1806.17	214.84	3025.31	**1805**	1805.88	286.72	4041.36	**1804**	1805.69	368.83	5205.18	**1804**
	± 0.81	± 36.90	± 527.53	(2%)	± 0.71	± 71.15	± 1013.19	(6%)	± 0.76	± 87.04	± 1237.96	(13%)	± 0.71	± 115.74	± 1650.78	(17%)	± 0.71	± 155.19	± 2207.03	(3%)	± 0.74	± 188.65	± 2678.75	(4%)
16	1805.70	70.58	876.78	**1804**	1805.31	128.84	1630.24	**1804**	1805.21	154.55	1963.10	**1804**	1805.06	202.06	2574.04	**1804**	1804.93	244.81	3121.67	**1804**	1804.87	280.67	3587.77	**1804**
	± 0.61	± 39.75	± 512.26	(2%)	± 0.56	± 83.42	± 1074.44	(5%)	± 0.53	± 103.32	± 1332.54	(6%)	± 0.49	± 131.72	± 1698.65	(9%)	± 0.50	± 160.94	± 2072.66	(16%)	± 0.46	± 186.64	± 2406.07	(18%)
Iogra400_12	2004.17	91.72	1043.40	**2003**	2003.71	160.81	1843.63	**2003**	2003.59	189.04	2169.46	**2003**	2003.51	214.20	2464.17	**2002**	2003.27	320.20	3701.25	**2002**	2003.18	368.10	4245.00	**2002**
	± 0.65	± 40.25	± 466.54	(12%)	± 0.55	± 83.82	± 975.76	(34%)	± 0.49	± 106.66	± 1239.58	(41%)	± 0.52	± 130.65	± 1521.51	(1%)	± 0.51	± 196.50	± 2286.21	(3%)	± 0.48	± 222.19	± 2575.38	(4%)
16	2005.79	106.70	1101.39	**2004**	2005.40	172.35	1796.18	**2004**	2005.23	219.70	2292.77	**2004**	2005.14	254.03	2656.12	**2004**	2005.00	314.39	3297.08	**2004**	2004.81	429.73	4519.61	**2003**
	± 0.57	± 46.73	± 495.54	(1%)	± 0.57	± 92.96	± 984.48	(4%)	± 0.56	± 129.30	± 1364.50	(7%)	± 0.53	± 148.66	± 1570.78	(8%)	± 0.55	± 193.59	± 2043.34	(15%)	± 0.50	± 258.14	± 2734.82	(1%)
Iogra450_12	2204.89	111.55	1056.49	**2204**	2204.37	207.81	1988.19	**2203**	2204.19	262.27	2515.77	**2203**	2204.09	298.45	2865.62	**2203**	2203.95	359.66	3459.60	**2203**	2203.81	442.77	4262.83	**2203**
	± 0.61	± 50.71	± 491.80	(25%)	± 0.58	± 101.64	± 983.44	(4%)	± 0.56	± 131.03	± 1267.36	(8%)	± 0.55	± 149.87	± 1450.84	(11%)	± 0.55	± 199.66	± 1934.31	(18%)	± 0.50	± 263.58	± 2553.43	(24%)
16	2207.33	134.72	1168.50	**2205**	2206.60	251.24	2199.24	**2205**	2206.48	286.60	2515.75	**2204**	2206.37	328.69	2887.83	**2204**	2205.99	510.40	4485.24	**2204**	2205.88	576.34	5078.53	**2204**
	± 0.75	± 50.90	± 448.08	(1%)	± 0.63	± 111.82	± 987.44	(5%)	± 0.69	± 127.72	± 1129.55	(1%)	± 0.70	± 158.97	± 1407.11	(1%)	± 0.61	± 239.51	± 2114.13	(1%)	± 0.60	± 271.73	± 2407.62	(1%)
Iogra500_12	2403.76	126.42	1012.87	**2403**	2403.34	219.27	1778.37	**2402**	2403.22	269.27	2183.85	**2402**	2403.13	313.73	2551.13	**2402**	2403.03	380.55	3097.69	**2402**	2402.98	423.45	3446.24	**2402**
	± 0.55	± 65.09	± 529.74	(30%)	± 0.55	± 123.73	± 1013.13	(4%)	± 0.52	± 156.40	± 1283.20	(5%)	± 0.50	± 188.61	± 1545.29	(7%)	± 0.48	± 240.19	± 1969.81	(10%)	± 0.45	± 280.60	± 2295.93	(11%)
16	2407.03	156.91	1163.73	**2405**	2406.37	280.29	2092.59	**2405**	2406.22	333.76	2497.30	**2404**	2406.09	394.91	2957.00	**2404**	2405.86	533.88	3995.28	**2404**	2405.75	608.58	4567.99	**2404**
	± 0.68	± 59.62	± 449.23	(2%)	± 0.72	± 133.52	± 1007.71	(8%)	± 0.70	± 160.94	± 1213.04	(1%)	± 0.66	± 196.88	± 1486.75	(1%)	± 0.63	± 269.81	± 2029.76	(2%)	± 0.62	± 325.71	± 2453.74	(2%)

chances to find high quality solutions. Furthermore, the algorithm is robust to changes in the structure of problem instances, since the deviation of generated solutions from the optimal one is less then 3% in all experiments when $N_{it} \geq 200$. Moreover, as the number of iterations N_{it} is increasing, the quality of solutions increases for all problem instance. The biggest change in the quality of reported solution occurs between $N_{it} = 100$ and $N_{it} = 200$. Another influence of stopping criterion concerns the variability of the reported solutions' quality. Deviations of $\bar{y}$ in Tables A.1 and A.2 indicate that the higher value of the N_{it} is decreasing the variability of the response value. The conclusion becomes more evident while observing difference between standard deviations of $\bar{y}$, obtained for $N_{it} = 100$ and $N_{it} = 200$. More specifically, let Iogra100_12 be the case study. The standard deviation has decreased 9 times while solution quality increased around 1.5%. Similary, the stabilization of variability can be detected for the rest of problem instances within this case study.

In Fig. A.5 the dependency of solution's quality on the value of stopping criterion is more obvious. Each graphic represents propagation of a mean of solutions' quality over different values of N_{it} for two instances with the same number of tasks and different m (Table A.1). It should be noted that the graphics are not scaled evenly. For each problem instance, a slow improvement of the solution's quality can be observed after N_{it} reaches 200. Another interesting conclusion drawn from these graphics concerns the impact of the number of machines on the hardness of instances of the corresponding scheduling problem. It seems that the impact is not the same for all problem instances and it depends on the maximal number of iterations. Two cases can be distinguished, when $N_{it} < 200$ and $N_{it} \geq 200$. In the first case, on all instances of the class $m = 16$, $\bar{y}$ obtains lower quality compared to quality of solutions reported for instances of class $m = 12$. However, when $n = 100$ and $n = 350$, even when $N_{it} \geq 200$, `sLPT+BF` heuristic generates better solutions on instances of class $m = 16$. This indicates that the two instances are easier to solve if more machines are included in the scheduling problem and if some minimum computational effort (number of iterations) is provided. However, the number of machines has a large impact on a running time. Reported values from Tables A.1 and A.2 show that, in average, more time is needed to generate complete solution for class $m = 16$. Moreover, in majority of the cases, problem class $m = 16$ is still more difficult to solve w.r.t. the quality of a solution.

In addition to mean values reported in Tables A.1 and A.2, in Appendix B we provide graphics containing frequency of occurrences of all objective function values within a single experiment of the `sLPT+BF`. We are interested in results when $N_{it} \geq 1000$. The graphics provide better perspective into the expected value of quality of the solution later used for analysis of BCO. Results are divided between two problem classes, and graphically presented for $m = 12$ in Figures B.1 and B.3, and for class $m = 16$ in Figures B.2 and B.4. From these charts we can conclude that the reported estimates in form of average values from Table A.1 are agreeing with the distribution of the frequencies of reported solutions. The reported mean values and their standard deviations represent a good statistics to describe the overall success of the `sLPT+BF` heuristic.

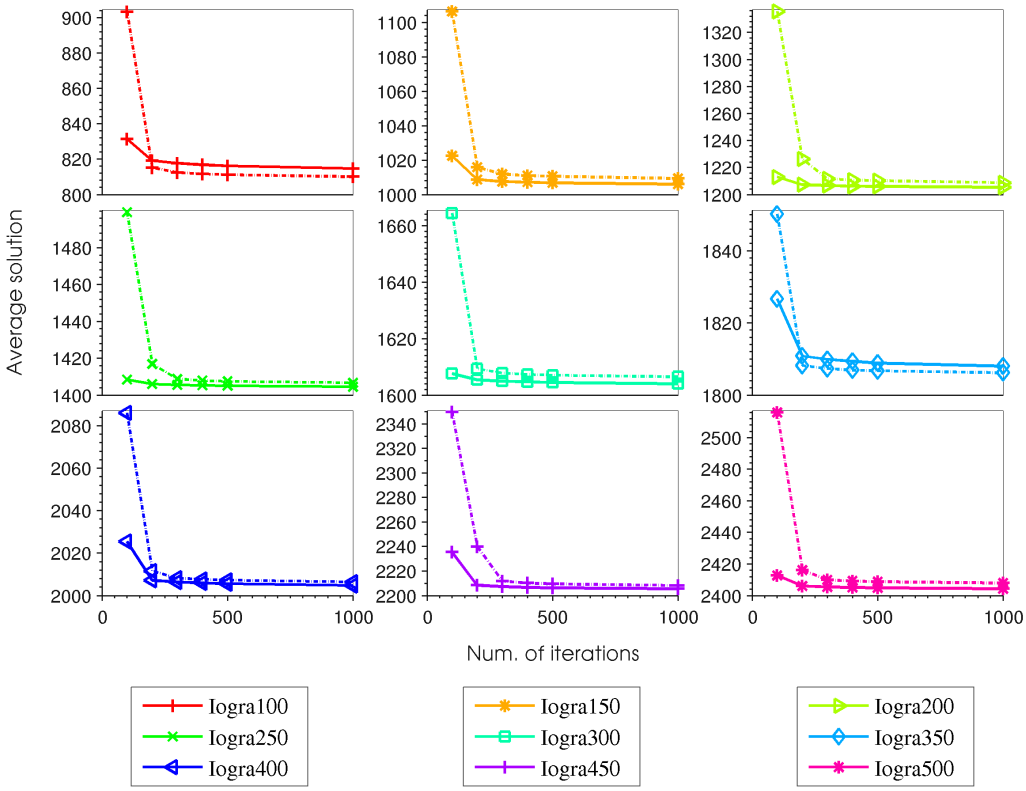


Figure A.5: Propagation of solutions for different instances generated by sLPT+BF after 100, 200, 300, 400, 500 and 1000 iterations. Solid line corresponds to class of instances when number of machines is 12, and dashed line to 16 machines.

A.2 The BCO algorithm

A.2.1 BCOc for $P||C_{max}$: calculate constructive moves

Parameter NC was used to determine number of alternations between forward and backward passes. Avoiding to define new parameter, the number of components that should be scheduled in one run of forward pass was controlled by variable n_task_all. The values of the variable are integer, determined for number ot tasks n following the rule:

$$n\_task\_all = \lfloor \frac{n}{NC} \rfloor.$$

However, when two numbers are not divisible the reminder is then greater than zero. For that reason the variable $rest$ is used to hold the values of the reminder of the division, i.e.,:

$$rest = n\%NC.$$

The counter n_task_all is utilized to control the number of tasks scheduled during each forward pass. Variable iii_count is initialized to n_task_all. If $rest > 0$ then

iii_count is incremented by 1 for the first $rest$ forward passes. The procedure states that tasks are being scheduled one by one until iii_count is reached. Pseudo-code of procedure in the BCO algorithm is presented in Fig. 1.

Algorithm 1 Procedure to calculate number of components for each forward pass in an iteration of BCOc.

Precondition: Read problem data and stopping criterion.

```
⋮
i ← 0;
n_task_last ← rest;
for (u = 0; u < NC; u++) do
    iii_count ← (u + 1) · n_task_last;
    if (n_task_last > 0) then
        iii_task_last ++;
        n_task_last −−;
    end if
    ▷ forward pass
    ⋮
    ▷ backward pass
    ⋮
end for
```

A.2.2 Choice of reference case: heuristic vs. $NC = 1$

We prove that the standalone heuristic can be represented by the *reference case* of BCOc algorithm (i.e., $NC = 1, B = 20$). To address this question we compare two sets of data: the first generated by `sLPT+bestfit`, and the second by the BCO algorithm with configuration $(NC = 1, B = 20, p_b^0, min(f_b^1))$. We chose this configuration during the experimental study of BCOc, soon as we noticed that all BCOc algorithms perform equally well for $NC = 1$ and $B \in [1, 20]$. This is also easy to establish without experiments since for $NC = 1$ backward pass is not employed. Therefore, qualitative parameters of BCOc are not able to impose their influence. Additionally, for maximal number of iterations and $NC = 1$ the best BCOc algorithm is acquired for the maximal population size ($B = 20$).

We had reasons to expect that the standalone heuristics might achieve better performance than the corresponding BCOc as the heuristic attains higher number of iterations that allows frequent update of the global best solution. Thus, it learns about the appropriate value for y_{max} that would lead towards high quality solutions. In particular, we set $N_{it} = 2000$ for `sLPT+bestfit` and $N_{it} = 100$ for the corresponding BCOc algorithm for which $B = 20$. We do not disregard the advantage coming from the population of solutions as the exploration component allows the search to start from different positions of the search space and, therefore, explore multiple of solutions before the start of new iteration.

The results of comparison are represented in Fig. A.6 as a graphic of propagation of the average best reported solution's quality, expressed as relative error, over problem size n. We conduct the comparison for all problem instances. From Fig. A.6 we observe

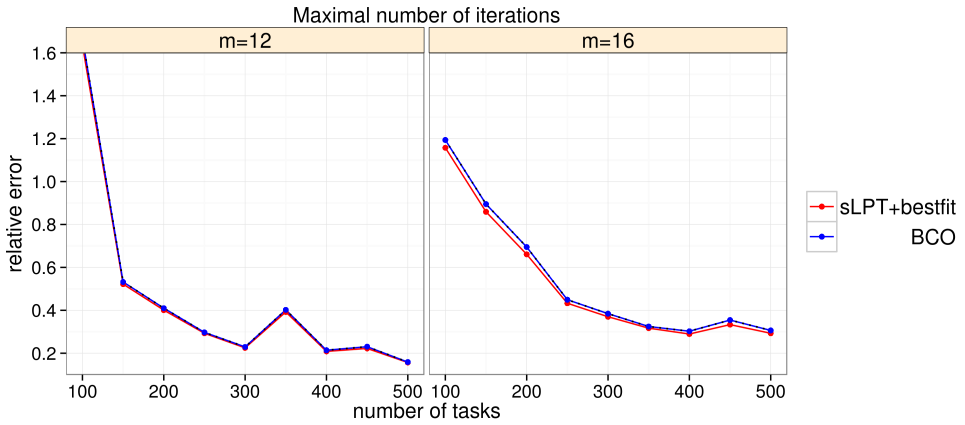


Figure A.6: Graphic of propagation of the average quality of solution generated by standalone heuristic and of BCO algorithm when $NC = 1, B = 20$. Stopping criteria is N_{it}.

that results do not differ significantly. We conclude that all comparisons between different parameter configurations also show performance against the standalone heuristic.

A.3 Statistical analysis of BCOc

Statistical tests are founded on one performance measure, i.e., solution quality. Furthermore, by controlling termination of BCOc by maximal number of iterations we do not account running times. Therefore, T does not influence the conclusions of the tests. We extend our study of parameter effects by restricting maximal allowed CPU time. New results are compared against results of N_{it} to capture any changes in our conclusions. Consequently, statistical analysis of BCOc is centralized in framework of structural tunning.

A.3.1 Preliminaries

A.3.1.1 The best mean

The best mean of a particular BCOc (for fixed qualitative parameters) refers to the smallest average quality of solutions among totally $B \cdot NC$. This value is reported in the basic table. In Section 8.3.4 we discuss about how to calculate the best mean in (Formula (8.2), pg. 154). We remark about its corresponding data sample. A *data sample* of the best mean is a set of solution quality values y_s produced by the BCOc instance w.r.t. $seed$.

A.3.1.2 Data sample for (non)parametric test

Parametric tests employed on the best means are based on the corresponding data samples, where a data sample represents a product of a single experiment. In particular, in the thesis the data sample consists of n_{run} measured outcomes generated by

the corresponding BCOc instance for different values of parameter $seed$ (Section 7.2). Because values of $seed$ are fixed by associating them with an index of a run within an experiment (execution counter), the data samples in our (non)parametric tests are paired (dependent). All the samples used in this thesis contain outliers. In particular, when the condition of normality is not satisfied we do not eliminate the outcomes that produce the skewness.

A.3.1.3 Statistical tests for paired data

Data samples that generate the best means are *paired* and the null hypothesis is the equivalence of means. During the statistical tests we are not interested in between-subject differences, caused by values of seed as nuisance factor. Consequently, the choice of statistical test is narrow, i.e., we may employ repeated measures ANOVA and/or Friedman's test[(1)]. The parametric test represents good alternative, under conditions of a normal distribution of a sample. Its advantage against the non-parametric test, such as Friedman's, is simplicity of obtaining size of effect.

A.3.1.4 Dominance of algorithm

Difference in the performance between two loyalty functions must satisfy condition that one has to show clear dominance for at least one configuration pair that belongs to $\mathcal{B} \times \mathcal{NC}$. Here, by dominance we imply that there exists at least one pair (i, j), $i \in \mathcal{B}, j \in \mathcal{NC}$ for which some loyalty function performed the best compared to others. Furthermore, compared against the best result of other loyalty function, there needs to be clear statistical difference between two purported results. Otherwise, statistical difference in the performance between a group of loyalty functions cannot be established. However, the dominance does not take into account sensitivity of loyalty functions to all values of parameters B and NC. Therefore, we are not interested for which i and j the best performance is established.

A.3.1.5 Effect of ME across the study

Here, we describe comparison between methods of evaluation on the complete problem set for $P||C_{max}$. According to solution quality landscapes, presented in Section 8.4 different levels of the parameter ME may occasionally produce high quality solutions. According to Fig. 8.3 the influence of the problem structure is evident. We will base our study on the best configurations of (Lp, B, NC). We conduct Friedman's rank test to compare three methods of evaluation. It is based on ranking the best representative among $Lp \cdot B \cdot NC$ reported solutions. The first line of the study is, therefore, to determine influence of parameter Lp, i.e., which loyalty function exhibits similarities across the group (within the level of ME).

For Friedman's rank test the dependent variable is method of evaluation, and problem instances an independent variables. To determine the ranks we utilize RMANOVA with fixed set of configurations as dependent variable and $seed$ as independent. Ordinarily, two-way ANOVA should be employed, however, normality assumptions are not satisfied for all levels of dependent variable. Calculation of ranks for each problem

[(1)] The utilization of *permutation test* has increased over the years, which is the motivation to include it in the future work.

instance is conducted by procedures of multiple comparison using RMANOVA to test the hypothesis of equivalence of means, and pairwise t-test with Hommel's correction as a post-hoc test. The procedure in R is shown in Section A.3.3.

A.3.2 Repeated measures ANOVA

The one-way repeated measures ANOVA is used to compare mean values for ordered type of results across multiple related (dependent) groups. The independent variable has categories called *levels*, which will be referred here as "groups". The results, or dependent variable, are compared between the categories of the grouping, or independent variable, which is referred to as the *within-subject factor*. Typical studies that involve RMANOVA consider changes of mean values over different time points, or when we want to establish differences between mean values under different conditions. The later approach was of interest in this dissertation. It is presumed that the data of the population to which statistical inference is to be made, follow normal distribution within each group. For k related groups the null hypothesis (H_0) is that the population mean μ of the measured results are the same for all of the related groups:

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_k.$$

The alternative hypothesis (H_a) states that k related populations means are not equal, i.e., that there is at least one mean that is different from others (there are least two samples with different mean values). The test does not reflect where the differences between groups occurred, as it represents an omnibus statistical test. Therefore, to investigate where exactly differences occurred, a post-hoc test has to be conducted.

A.3.2.1 RMANOVA in the BCOc study

To employ RMANOVA in our study we need to satisfy basic conditions: 1. dependency between samples; 2. the number of data in the sample is large enough for parametric test; 3. symmetric distributions; 4. sphericity condition. The first condition is known as *paired data samples* and describes the relationship among response values generated by different BCO instances. Since the solutions are discrete, usually, during an experiment, same solution can be generated by different seed. This causes generation of *binned data*, which in this case corresponds to grouping of data of the same value. The frequency distribution of the data is a good representative when the mean of the frequency distribution of the sample coincides with the average value of the original sample.

A.3.2.2 RMANOVA of Lp groups

In this section only the main effects of the statistical tests are being presented. Reason why RMANOVA is utilized is because there is no interest in the variance cause by parameter $seed$, i.e., between-subject difference. RMANOVA test separates the effect caused by the subject ($seed$) from the errors which we want to investigate. The distribution of the response values in groups is presumed to be normally distributed, based on graphical assessment as presented in Fig. A.7. The RMANOVA tests the *within-subjects* effects, i.e., if significant difference between the means generated by loyalty functions exists.

After concluding an overall (in)significant difference in means, a series of post-hoc tests is conducted in order to find out which of the loyalty functions caused difference.

A.3.2.3 Distribution of samples: normality test

To examine condition before applying RMANOVA test, normality tests is used. In order to provide meaningful model of the sample, frequency distribution (histogram) is chosen. Such representation is suitable to represent discrete data, as is here the case. Moreover, one may recognize if the data sample holds some central tendency or follows some known distribution, as detect gaps and outliers. First test of normality is conducted by graphical representation of samples distribution, provided in Fig. A.7. The graphics represent relative frequencies[(2)] of results for one case. The chosen collection of results is generated when BCO parameters are set to $min(ev_1)$, $p_b^i, i \in [0,9], i \in \mathbb{Z}$, $N_{it} = 100$ and problem instance to *Iogra100_16*. In other words, each graphic denotes a case of comparison between two samples of data generated during an experiment by two different loyalty functions within same method of evaluation and on same test instance. To remind, a sample of data describes a set of y_{max} solutions generated by mentioned parameter configurations and the best reported configurations of B and NC during an experiment. Histograms in Fig. A.7 are produced with `R` package language, function `hist`, adjusted in order to show right-closed intervals (`right=TRUE`) and manually setting break points on x-axes in order to provide suitable interval for both graphics. Due to overwhelming number that would take place when comparing all pairs of loyalty functions, the best performing p_b (blue), is compared to the rest from the group colored with green. For each histogram, a mean of a sample is provided as $\overline{y_{max}}$. The source of variability of each sample is caused by parameter $seed$.

The Mk1 Eyeball test[(3)] revealed a good fit to normal distribution in the histograms made in `R`. Furthermore, the skeweness of the groups are in the acceptable range. In addition, homogeneity of variances was analyzed by calculating ratios of largest to smallest variation. Therefore, in Fig. A.7 all samples are model by normal probability curve defined by the average value of the sample and its standard deviation, and colored in respect to color of the sample histogram. The visual assessment is that data distribution resemble the normal distribution.

To complement the graphical assessment of the normality, a goodness of fit with normal distribution was tested with Shapiro-Wilk test. The null hypothesis of Shapiro test is that a sample is normally distributed. Measured p values on all samples failed to verify normal distribution. However, that might be since the standard Shapiro-Wilk test for normality is not suitable for binned data [Roy82]. Another important feature of parametric tests is the test of equality of variance between the data samples. Because of that Leven's non-parametric test was used, however didn't always verify the equivalence of variance between considered samples. When it failed, differences between variance of each of the sample was compared and it was shown not to be larger than 4 ([How10, pg. 334]) in majority of cases. Where the difference in variance is larger in shown in Fig. A.7 for p_b^6. This sample has an outlier, which causes large skewness in the data set. However, the number of outliers is very small, and the data shows clear grouping around the mean.

(2) *Relative frequencies* may be generated when normal frequencies are normalized.
(3) Mk1 is abbreviation of *Mark One Eyeball* used in a military as a slang for visual inspection [Bus].

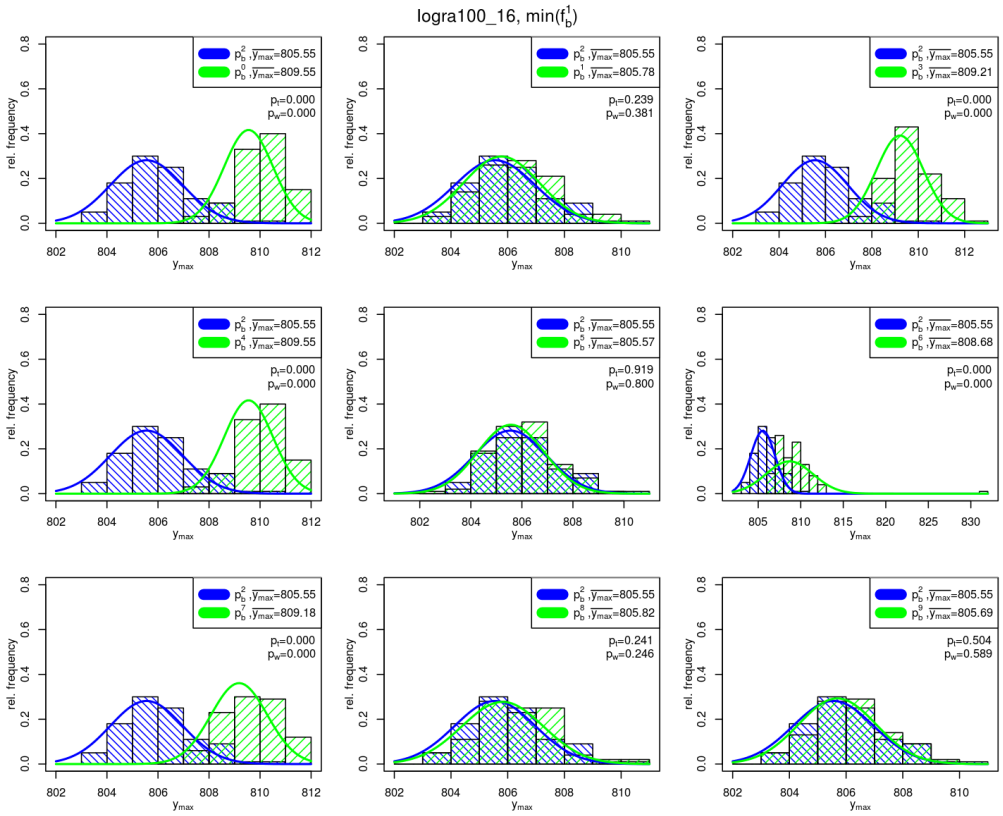


Figure A.7: Pairwise comparison between results of the best performing loyalty function (here p_b^2), and the rest of loyalty functions within the same group, on test instance Iogra100_16.

Fig. A.7 presents, beside the assessment of normality, p-values of two different statistical tests, i.e., p_t for parameteric *t-test* and p_w for non-parametric *Wilcoxon test*. Tests are conducted on the given pairs of samples with `R` package[(4)]. By comparing p-values between the tests we confirm that parametric tests are suitable to conduct empirical evaluation of the BCOc algorithm.

Furthermore, results in Fig. A.7 show that employing RMANOVA provides reliable conclusions of our hypothesis testing. To conduct RMANOVA test we use ezANOVA package [Law15]. The condition of sphericity is integrated inside of the calculation of p-value. Sphericity condition can be expressed in various ways [Dav02, pg. 109]. In general, sphericity refers to condition where the variances of the differences between data groups are equal [Dav02]. When conditions are not satisfied, corresponding corrections are made inside of ezANOVA function. All this indications are reported in our study, especially if sphericity condition is not satisfied, as we mark it with the reference to the type of corrections.

[(4)] Default setting of t-test assumes unequal variance and applies the Welsh modifications.

A.3.2.4 Cohen's table

Cohen's table for different effects for within-subjects comparison is: small (η^2 = 0.0099), medium (η^2 = 0.0588), and large (η^2 = 0.1379) effects [Coh88, pg. 285]. When $\eta^2 = 0.009$ about 1% of the total superpopulation variance accounted for by group membership. This values are used only as guidance of estimating the size of effect. Usually, the Cohen's table is used as a last resort, and the advice is to compare calculated effect size against other effects from the literature or along the study.

A.3.3 Ranking R procedure for ME

We conduct multiple comparison using RMANOVA and generate p-value and distinguished three p-values: p_1, p_2, p_3. Value p_1 describes comparison of a pair $\min(ev_1) - \max(ev_1)$, p_2 of a pair $\min(ev_1) - \max(ev_2)$, and p_3 of a pair $\max(ev_1) - \max(ev_2)$. If the result of RMANOVA is $p >= 0.05$, all three ranks are appointed to value of 2. If $p < 0.05$ pairwise t-test with Hommel's correction is used in order to capture statistical difference between pairs of methods.

The ranking procedure is based on three samples (one sample includes n_{run} response values). A mean value of each sample coincides with the best result produced on configuration sub-space $\mathcal{L} \times \mathcal{B} \times \mathcal{NC}$ at each level of ME. To prepare values for the ranking procedure, we presume three input files, each for different levels of ME. One file contains at most 10 columns, each column representing solution quality data sample generated at one level of Lp (we presume that columns are organized in an increasing order along indicies of Lp). The first step is to extract from each file the column with the smallest mean value. The three new columns form matrix M. We call `R` function `stack` on M to prepare for call of `ezANOVA` function that conducts RMANOVA test. The procedure `bco-stack` is described in pseudo-code Alg. 2. The ranks are calculated following the procedure of Alg. 3. Firstly, simple ranks are determined with the `R` function `rank`. In case of ties (two means report similar values) established by RMANOVA test, we average the ranks. In our study the ties occur in different occasions w.r.t. result of RMANOVA test. The first, and the most evident is for $p \geq 0.05$ at the significance level $\alpha = 0.05$. The ranks $r_i, i \in \{1, 2, 3\}$ are equal to 2. In case $p \leq 0.05$ we conduct pairwise test with Hommel's correction and observe three reported p-values. The ties may occur if two p-values have not detect significant differences. Here, we

Algorithm 2 Transform data available as separate columns in a data frame.

Precondition: Input file with values of matrix M

```
function BCO-STACK(M)
    ncol ← length(M);
    M.aux ← cbind(as.data.frame(1:length(M[, 1])), M);
    colnames(M.aux) ← c("first");
    M.stack ← stack(M);
    subject ← rep(M.aux$first, ncol);
    M.stack[3] ← subject;
    colnames(M.stack) ← c("values","main","subject");
    return M.stack
end function
```

Algorithm 3 Calculating Friedman's ranks.

Precondition: Three input files $me_i \in ME$, library(ez)

```
function POST-HOC(me_1, me_2, me_3)
    M ← matrix with three columns representing the best from each me_i;
    M.s ← BCO-STACK(as.data.frame(M));
    a.mean ← apply(M.s, 2, mean);
    a.rank ← rank(a.mean);
    Meza ← ezANOVA(M.s,dv=.(values),wid=.(subject),within=.(main) ,detailed=TRUE);
    ef ← unlist(Meza);
    p.value ← as.numeric(ef[14]);
    if p.value < 0.05 then
        p.a ← with(M.s,pairwise.t.test(values,main, p.adjust.method ="hommel",paired=T))
        p.u ← unlist(p.a);
        if (as.numeric (p.u["p.value1"]) ≥ 0.05) then
            a.rank[1] ← (a.rank[1] + a.rank[2]) / 2;
            a.rank[2] ← a.rank[1];
        end if
        if (as.numeric (p.u["p.value2"]) ≥ 0.05) then
            a.rank[1] ← (a.rank[1] + a.rank[3]) / 2;
            a.rank[2] ← a.rank[1];
        end if
        if (as.numeric (p.u["p.value4"]) ≥ 0.05) then
            a.rank[1] ← (a.rank[2] + a.rank[3]) / 2;
            a.rank[2] ← a.rank[2];
        end if
    end if
end function
```

average the ranks among the two mean values that have reported the largest p-value.

A.4 Empirical study of BCOi

A.4.1 SATLIB

The initiative to create SATLIB library started in 1998 as a co-joint work of Hoos and Stützle [Hoo00b]. According to [Hoo00b] a motive to create SATLIB library originates from an increased interest in the experimental studies of SAT solvers. The goal is to help assessment of new algorithmic approaches. Furthermore, SATLIB is created to support the constant flow of new benchmark problem instances and to serve as dynamic environment of generating intrinsically hard SAT problems.

A.4.1.1 Categorization

Among the first to propose procedures that would distinguish randomly generated hard problems from easy ones are Mitchell, Delman and Levesque [Mit92]. Their work has been inspired by researchers who claimed that SAT problem can be solved in average in $\mathcal{O})(n^2)$ steps (see *Introduction*, pg. 1). The authors discuss significance of problem categorization and suggest procedures that generate hard instances. Difficulty of

a particular problem instance was measured with Davis Putnam (DP) procedure. Two models of instance distributions are distinguished: 1. fixed clause-length model, and 2. constant-probability model. [Mit92] conclude that fixed clause-length model generates harder instances if the number of clauses is roughly 4.3 times larger than the number of variables, whereas formulas with either more or fewer clauses are easy. Consequently, generating larger formulas does not necessarily lead to harder formulas. Prior to their work [Che91] categorizes SAT instances as *under-constrained* and *over-constrained*, which respectively describes formulas that have fewer clauses and formulas with many clauses. [Mit92] conjecture that formulas, known as *critically-constrained*, are much harder as they have much less satisfying assignments. They conclude that the probability that a random SAT problem instance is satisfiable is related to the ratio of number of clauses to number of variables (*clause-to-variable* ratio, α).

A.4.2 Number of transformations

The auxiliary information, omitted in the main material about BCOi does not uphold as much text as for BCOc. Here, we give description for BCOi of computing values of u (forward/backward pass counter) and of NCT in the last iteration while dealing with SATLIB 3-SAT problem instances. The input data for each problem instance is MAXFLIPS, NCT and a threshold value of parameter NC.

Since NC is not utilized as a control parameter that directs the search in the forward pass, to control values of variable u we first determine the values of iteration counter n_{iter}. Then,

$$u = mod(n_{iter}, NC).$$

We appoint $u = NC$, when u reaches value 0.

Moreover, we implemented BCOi so that the stopping criterion is defined as either solving the 3-CNF formula, or reaching $MAXFLIPS/(NCT * B)$ iterations. The value of NCT in the last iteration needs to be calculated for all the bees, in case that $NCT * B$ does not divide MAXFLIPS. If/when the algorithm satisfies the last condition, we check if $MAXFLIPS/(NCT * B) = 0$. If satisfied, the algorithm finalizes. Otherwise, the rest of transformations NCT_{rest} as determined by:

$$NCT_{rest} = (MAXFLIPS\%(NCT * B))/B.$$

The procedure allows equal distribution of workload between the bees, however, permits the BCOi algorithm to obtain more/less transformations than MAXFLIPS, which we refer to as *missed* transformations. Because in the study we employ small number of bees, the number of missed transformation is of practical significance.

APPENDIX B

Tables and graphics

B.1 Empirical study of BCOc

B.1.1 Analysis of the bestfit heuristic

Here, we provide additional graphics to complement conclusions from section A.1.5.1. The graphics in Figure Fig. B.3 and B.4 represent occurrence of best found solutions within 100 repetitions generated by `sLPT+BF` heuristic for nine problem instances of the same class (same number of machines). To observe distributions of the solutions for larger number of iterations, we used $N_{it} = 1000$ and $N_{it} = 10000$.

B.1.2 BCOc: response landscape for two methods of evaluation

In this section we provide response landscapes of BCOcs generated over domains of parameters B and NC for problem instance $Iogra100\_12$. In Fig. B.5 The figure shows influence of Lp within group $\max(ev_1)$. In figure we detect a large range in solution quality generated by particular values of parameter Lp over complete domains of quantitative BCOc factors. However, the algorithms generate high quality solutions for small NC and, the most often, for larger values of B. In Fig. B.6 the range between the smallest and the largest response value is smaller than in previous case. However, the method of evaluation produces high quality solutions for same NC as earlier, i.e., when it takes smaller values. Additionally, B should also be larger when the corresponding method of evaluation is coupled with the loyalty function which produce highly non-linear surfaces. Coupled with Fig. 8.2 we may state that the method of evaluation is the most influential parameter as it generates the largest discrepancy between the best and the worst response value over the rest of the BCOc parameters.

B.1.3 Comparison of 3-D surface plots

In Fig. B.7 the four differently colored response surfaces correspond to solution landscape generated by four BCOcs for problem instance *Iogra100_12*. In particular, method of evaluation $\min(ev_1)$ with loyalty functions $p^{0,u}, p^{1,2,8}$.

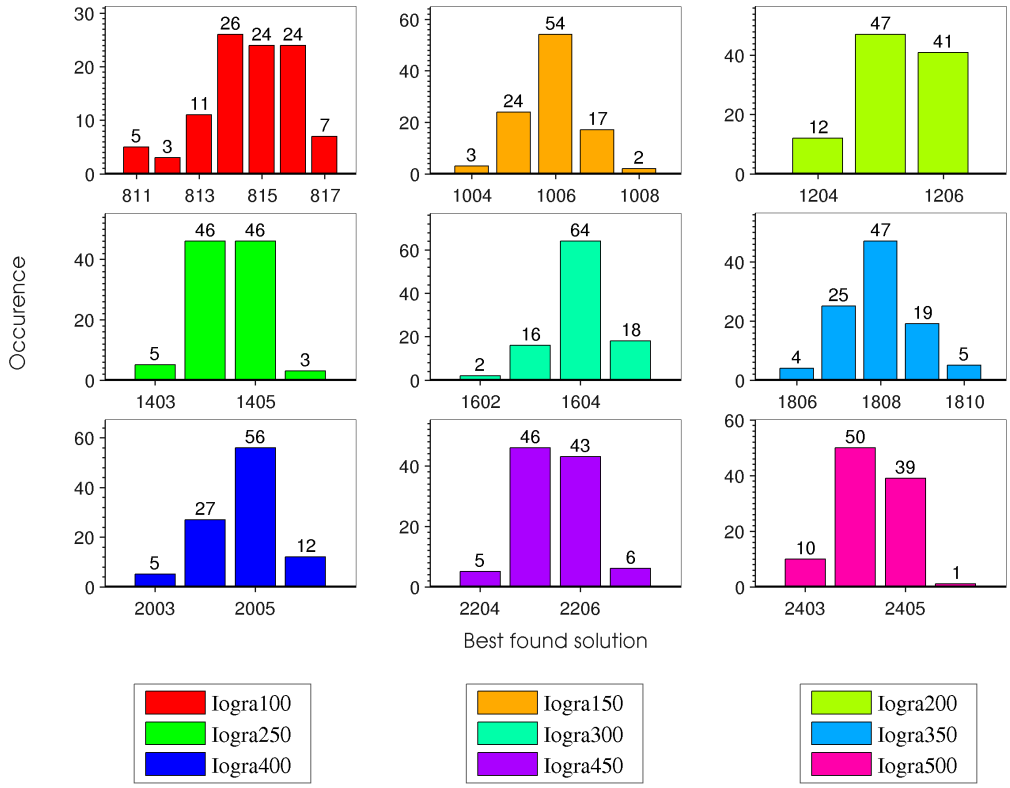


Figure B.1: Occurence of best found solution of each run generated with `sLPT+BF` for different instances, when $N_{it} = 1000$, $m = 12$.

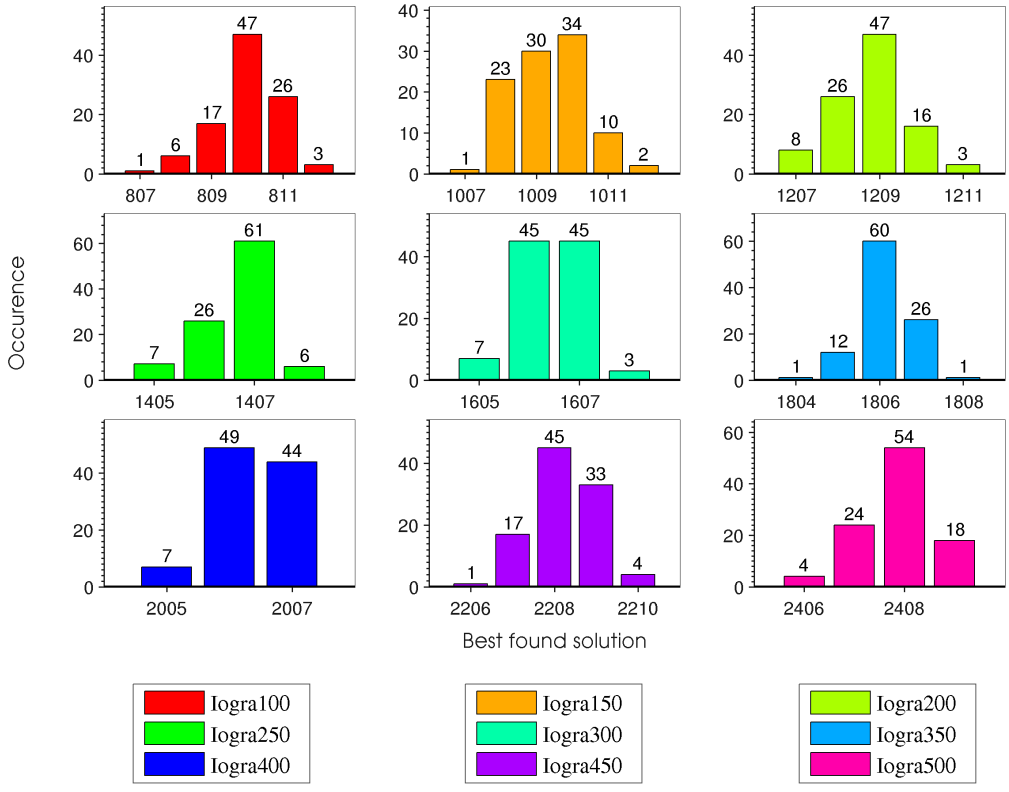


Figure B.2: Occurrence of best found solutions of each run generated with *sLPT+bestfit* for different instances, when $N_{it} = 1000$, $m = 16$.

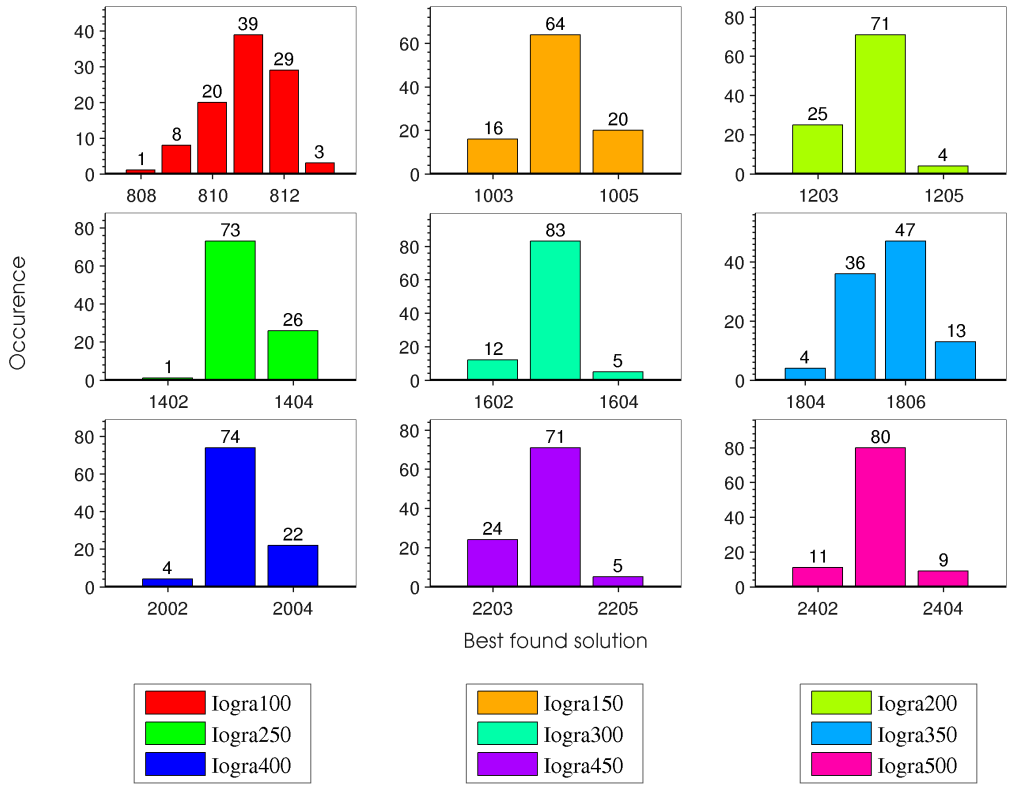


Figure B.3: Occurence of best found solution of each run generated with `sLPT+BF` for different instances, when $N_{it} = 10000$, $m = 12$.

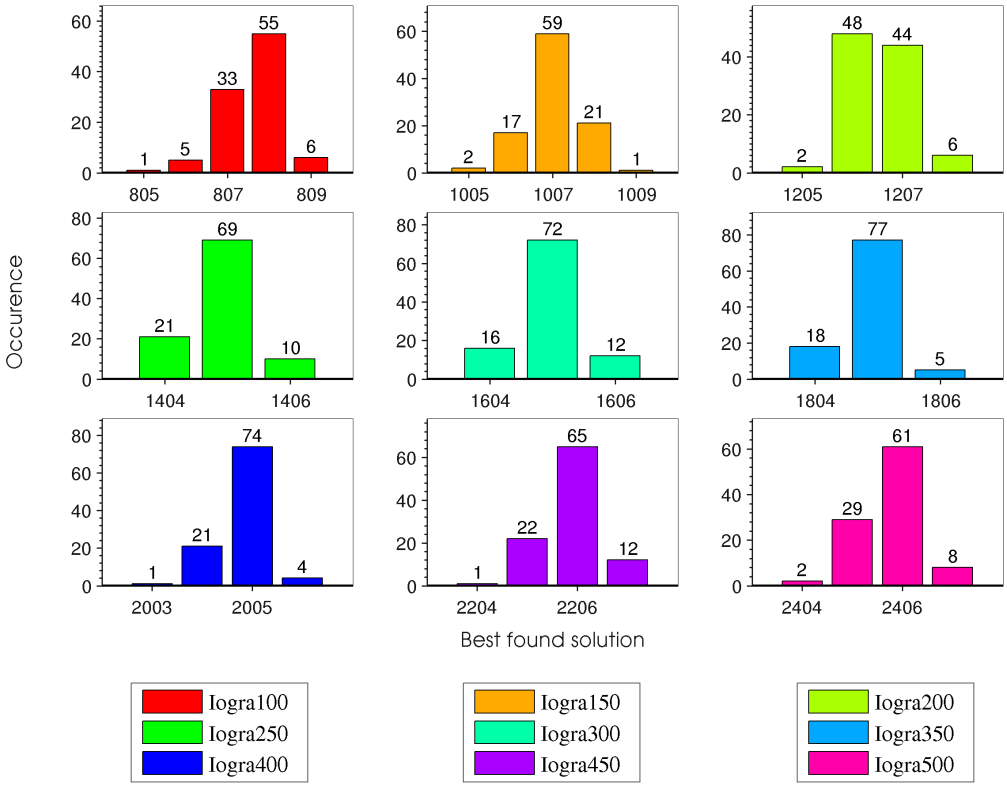


Figure B.4: Occurrence of best found solutions of each run generated with *sLPT+bestfit* for different instances, when $N_{it} = 10000$, $m = 16$.

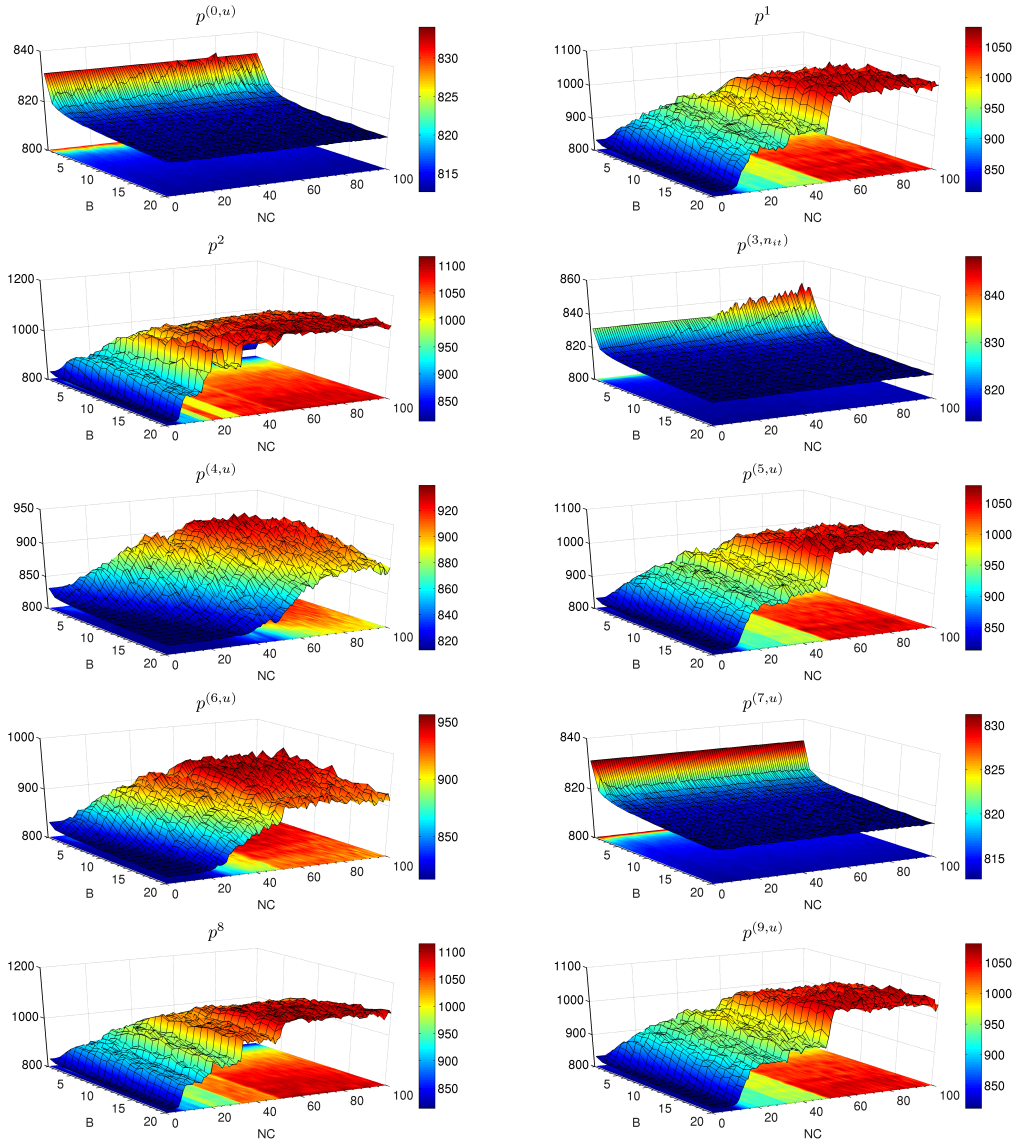


Figure B.5: Response landscape of 10 BCOc with $\max(ev_1)$ for Iogra100_12, $N_{it} = 100$.

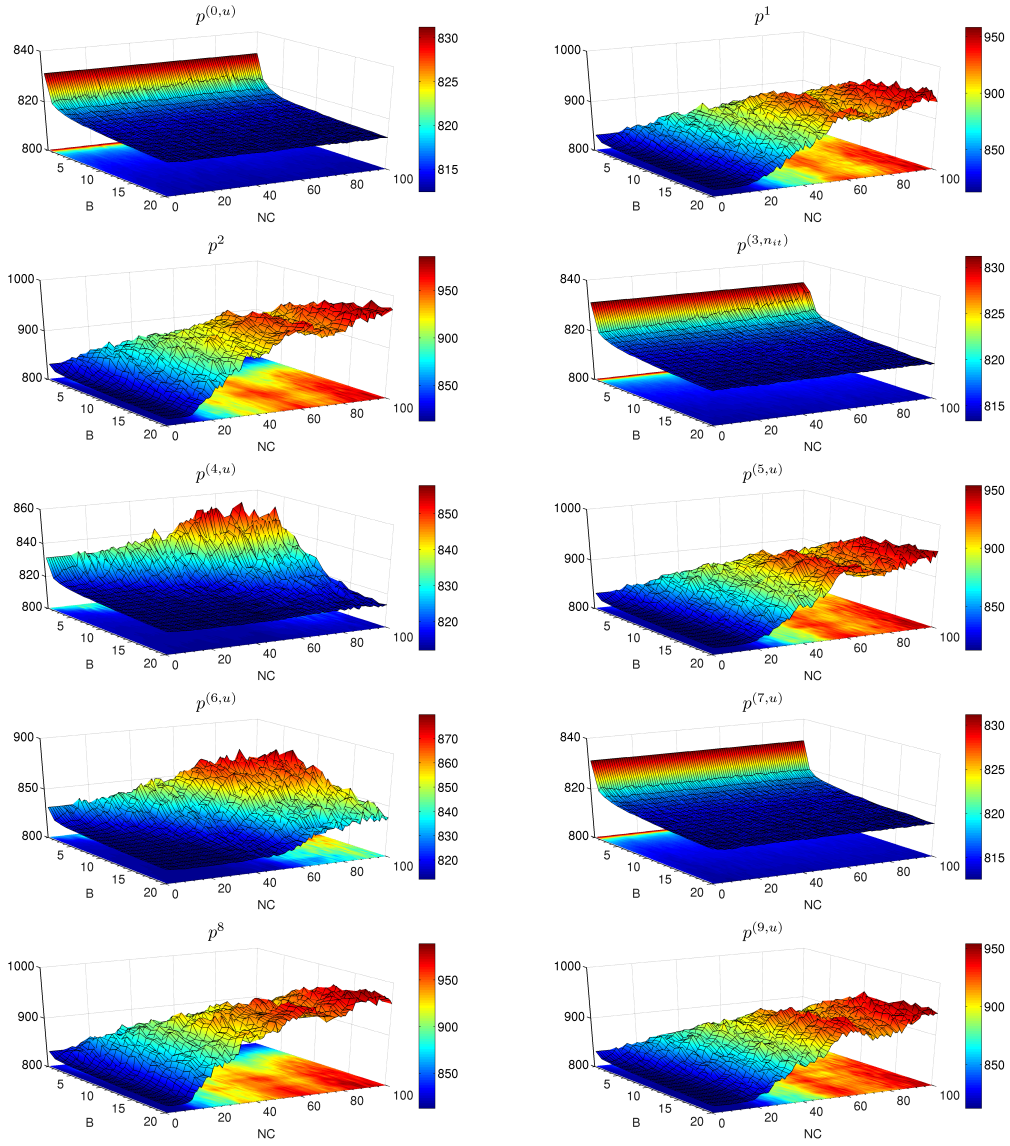


Figure B.6: Response landscape of 10 BCOc with $\max(ev_2)$ for Iogra100_12, $N_{it} = 100$.

B.1.4 Case study: Maximal allowed CPU time

The results of the RMANOVA test for equivalence of means between loyalty functions, conducted for each level of method of evaluation, are presented in Table B.1. Conclusions regarding results of the table are given in the main section of the dissertation (pg. 175). Table is organized as follows. The first column identifies the type of a problem instance. The table is split between three main groups identified with parameter ME. Result of a test is p-value, thus, we distinguish the results of RMANOVA and Friedman's test. A size of effect is given under η_g^2.

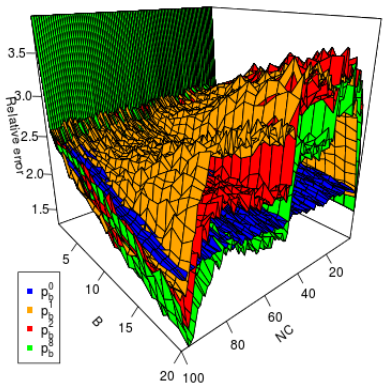


Figure B.7: Landscape of measured outcomes for four different loyalty functions, when $N_{it} = 100$, $min(ev_1)$ and problem instance *Iogra100_12*.

B.2 Tables and Figures

In this section we give auxiliary graphics and tables that were omitted in the main sections of this thesis. All the presented tables concern results of experimental study conducted for BCOc on $P||C_{max}$. We distinguish two broad groups of this results by the type of stopping criteria, i.e., N_{it} and T. The graphics at the end of this section represent bar chars of the problem instances tasks $P||C_{max}$. Therefore, we can observe the frequency of occurrences of tasks with the same running times.

B.2.1 Performance table

In this section we present results of experimental study of BCOc on problem $P||C_{max}$. The tables are described in Chapter 8. Two groups of results are distinguished w.r.t. stopping criteria. Each table holds results for specific problem instance.

B.2.1.1 Iterations as a stopping criteria

We provide table of results when stopping criteria is maximal number of iterations. We start wtih Iogra150_12, followed by Iogra150_16. Therefor, we provide first for $m = 12$ and next for $m = 16$, interchanging for different problem size.

B.2.1.2 Time as a stopping criteria

The maximal allowed running time, as the stopping criteria, may provide a more fair comparison between different BCO instances. The running time was set to different values, depending on the number of tasks to be scheduled. Namely, instances with greater number n get more time. For example, for $n = 100$ and $m = 12, 16$, the stopping criteria was set to $0.1[s]$. In case $n = 150$ the running time was set to $0.15[s]$. The largest value of the stopping criteria w.r.t. to the time of a execution was required for $n = 500$, where $T = 0.5[s]$.

Table B.1: Repeated-measure ANOVA and Friedman's test results for equivalence of means between loyalty functions of specific method of evaluation for $\alpha = .05$ and maximum CPU time.

	$\min(ev_1)$							$\max(ev_1)$							$\max(ev_2)$						
Problem	$p[sph]$	$\hat{\epsilon}$	F	p	η_p^2	η_g^2	$p[f]$	$p[sph]$	$\hat{\epsilon}$	F	p	η_p^2	η_g^2	$p[f]$	$p[sph]$	$\hat{\epsilon}$	F	p	η_p^2	η_g^2	$p[f]$
Iogra100_12	0.000	0.47[1]	65.71	0.000[1]	0.40	0.35	0.000	0.357	0.91	45.69	0.000	0.32	0.30	0.000	0.101	0.89	26.95	0.000	0.21	0.20	0.000
16	0.000	0.68[1]	156.59	0.000[1]	0.61	0.57	0.000	0.262	0.90	6.97	0.000	0.07	0.06	0.000	0.449	0.91	8.24	0.000	0.08	0.07	0.000
Iogra150_12	0.000	0.68[1]	52.58	0.000[1]	0.35	0.31	0.000	0.456	0.91	24.90	0.000	0.20	0.18	0.000	0.953	0.94	2.09	0.028	0.02	0.02	0.083[5]
16	0.000	0.66[1]	88.16	0.000[1]	0.47	0.42	0.000	0.572	0.92	10.43	0.000	0.10	0.09	0.000	0.923	0.93	8.10	0.000	0.08	0.07	0.000
Iogra200_12	0.000	0.72[1]	128.73	0.000[1]	0.57	0.53	0.000	0.010	0.95[2]	25.61	0.000[2]	0.21	0.19	0.000	0.127	0.89	0.95	0.482	0.01	0.01	0.368
16	0.000	0.77[2]	40.36	0.000[2]	0.29	0.23	0.000	0.518	0.91	22.27	0.000	0.18	0.17	0.000	0.244	0.90	2.75	0.004	0.03	0.02	0.004
Iogra250_12	0.000	0.52[1]	2.82	0.095[1]	0.03	0.01	0.063	0.468	0.91	20.41	0.000	0.17	0.16	0.000	0.453	0.91	0.50	0.878	0.00	0.00	0.833
16	0.000	0.93[2]	52.02	0.000[2]	0.34	0.31	0.000	0.439	0.91	14.41	0.000	0.13	0.12	0.000	0.667	0.92	6.31	0.000	0.06	0.05	0.000
Iogra300_12	0.000	0.70[1]	2.36	0.061[1]	0.02	0.02	0.030[6]	0.025	0.97[2]	19.08	0.000[2]	0.16	0.15	0.000	0.328	0.90	0.48	0.890	0.00	0.00	0.901
16	0.000	0.93[2]	15.78	0.000[2]	0.14	0.12	0.000	0.432	0.92	20.09	0.000	0.17	0.15	0.000	0.496	0.91	2.20	0.020	0.02	0.02	0.051[5]
Iogra350_12	0.000	0.18[1]	10.47	0.001[1]	0.10	0.03	0.000	0.608	0.92	62.36	0.000	0.39	0.36	0.000	0.059	0.89	2.11	0.027	0.02	0.02	0.005
16	0.000	0.51[1]	5.52	0.020[1]	0.05	0.03	0.003	0.781	0.92	15.93	0.000	0.14	0.13	0.000	0.050	0.96[2]	0.59	0.797[2]	0.01	0.01	0.756
Iogra400_12	-	1.00[3]	0.82	0.368[3]	0.01	0.01	0.325	0.332	0.91	27.23	0.000	0.22	0.20	0.000	0.798	0.93	1.83	0.060	0.02	0.02	0.051
16	0.000	0.70[1]	67.76	0.000[1]	0.41	0.36	0.000	0.552	0.91	17.22	0.000	0.15	0.14	0.000	0.031	0.97[2]	0.62	0.778[2]	0.01	0.01	0.664
Iogra450_12	-	1.00[3]	0.63	0.428[3]	0.01	0.00	0.696	0.690	0.92	43.28	0.000	0.30	0.28	0.000	0.901	0.93	1.65	0.098	0.02	0.01	0.131
16	0.000	0.54[1]	4.48	0.034[1]	0.04	0.02	0.015	0.120	0.89	34.54	0.000	0.26	0.24	0.000	0.717	0.92	2.64	0.005	0.03	0.02	0.013
Iogra500_12	0.000	0.71[1]	46.79	0.000[1]	0.32	0.29	0.000	0.376	0.90	22.63	0.000	0.19	0.17	0.000	0.227	0.90	0.59	0.804	0.01	0.01	0.802
16	0.000	0.74[1]	0.53	0.537[1]	0.01	0.00	0.309	0.192	0.88	26.01	0.000	0.21	0.18	0.000	0.394	0.91	1.15	0.325	0.01	0.01	0.603

$p[sph]$ - significance (p-value) of Mauchly's statistic test of sphericity;
$\hat{\epsilon}$ - Greenhouse-Geisser estimator of sphericity;
$p[f]$ - significance (p-value) of Friedman's test;

[1] When sphericity condition is violated ($p[sph] < 0.05$) proposed Greenhouse-Geisser estimator is used for correcting p-value of repeated-measures ANOVA test [Dav02];
[2] When sphericity condition is violated ($p[sph] < 0.05$) and $\hat{\epsilon} > 0.75$, Huynh and Feldt estimator $\tilde{\epsilon}$ is used for correcting p-value of repeated-measures ANOVA test [Dav02], p. 111.
[3] '-' When only two data groups differ sphericity condition is always satisfied.
[4] '—' signifies that all outcomes were identical.
[5] Friedman's test failed to reject null hypothesis of equal means for $\alpha = 0.05$. However, repeated-measures ANOVA reported significant effect of Lp on solution quality.
[6] Repeated-measures ANOVA failed to reject null hypothesis of equality of means for $\alpha = 0.05$. Friedman's test reported significant effect of Lp on solution quality.

Table B.2: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra150_12* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	1005.32	3	20	1	61.97 ±21.39	25.20 ±8.72	40.50 ±1.23	**1004.85**	4	20	71	63.46 ±20.36	42.30 ±13.80	66.80 ±1.14	1005.01	3	20	50	60.67 ±21.32	37.00 ±13.10	60.90 ±1.06
	w	1022.35	127	1	1	94.46 ±9.04	1.68 ±0.49	1.77 ± 0.42	1022.35	127	1	1	94.46 ±9.04	1.61 ±0.53	1.71 ± 0.47	1022.35	127	1	1	94.46 ±9.04	1.78 ±0.44	1.80 ± 0.42
p_b^1	b	1005.32	3	20	1	61.97 ±21.39	25.10 ±8.58	40.40 ±1.19	1005.03	3	20	5	65.79 ±17.55	28.30 ±7.44	43.10 ±1.10	1005.11	4	20	7	62.67 ±19.14	28.20 ±8.55	45.00 ±1.12
	w	1022.35	127	1	1	94.46 ±9.04	1.74 ±0.46	1.80 ± 0.45	1082.71	206	5	88	98.15 ±5.34	12.50 ±0.94	12.70 ± 0.55	1029.52	128	2	70	93.36 ±10.13	6.36 ±0.81	6.81 ± 0.42
p_b^2	b	1005.22	17	20	74	78.98 ±17.46	62.20 ±13.80	78.60 ±1.25	1005.09	3	20	4	68.21 ±16.51	28.80 ±6.98	42.20 ±1.05	1005.05	4	19	4	66.80 ±19.02	26.90 ±7.56	40.10 ±0.83
	w	1022.35	127	1	1	94.46 ±9.04	1.64 ±0.48	1.73 ± 0.44	1111.86	252	15	89	99.27 ±2.36	39.10 ±1.28	39.40 ± 0.98	1029.52	138	2	19	94.10 ±9.45	4.25 ±0.64	4.55 ± 0.52
p_b^3	b	1005.32	3	20	1	61.97 ±21.39	25.00 ±8.54	40.20 ±1.21	1005.20	4	20	42	68.90 ±18.68	38.70 ±10.50	56.20 ±1.02	1005.24	4	20	4	60.21 ±21.44	25.60 ±9.22	42.40 ±1.13
	w	1022.35	127	1	1	94.46 ±9.04	1.74 ±0.46	1.80 ± 0.42	1036.17	133	2	99	97.38 ±5.71	5.52 ±0.68	5.69 ± 0.52	1022.35	127	1	1	94.46 ±9.04	1.75 ±0.46	1.83 ± 0.40
p_b^4	b	1005.32	3	20	1	61.97 ±21.39	25.20 ±8.62	40.40 ±1.23	1004.92	2	20	8	67.38 ±18.78	30.10 ±8.29	44.70 ±1.12	**1004.99**	4	20	7	65.78 ±20.30	29.50 ±9.09	44.80 ±1.13
	w	1022.35	127	1	1	94.46 ±9.04	1.66 ±0.51	1.76 ± 0.45	1036.72	113	3	86	95.32 ±9.54	7.69 ±0.91	8.04 ± 0.51	1027.41	91	2	89	94.28 ±9.92	6.37 ±0.81	6.74 ± 0.50
p_b^5	b	1005.32	3	20	1	61.97 ±21.39	24.90 ±8.64	40.10 ±1.23	1004.98	4	20	8	76.40 ±16.24	34.20 ±7.34	44.80 ±1.09	1005.18	4	20	6	64.80 ±22.40	28.80 ±9.82	44.10 ±1.30
	w	1022.35	127	1	1	94.46 ±9.04	1.70 ±0.48	1.82 ± 0.38	1079.10	243	10	95	98.04 ±4.72	26.50 ±1.41	27.00 ± 0.84	1028.38	119	4	60	92.02 ±13.20	12.70 ±1.92	13.80 ± 0.61
p_b^6	b	1005.32	3	20	1	61.97 ±21.39	24.90 ±8.45	40.20 ±1.26	1004.92	3	20	5	64.80 ±19.92	28.50 ±8.64	44.00 ±1.04	1005.01	3	20	8	64.02 ±19.22	29.90 ±8.91	46.50 ±1.00
	w	1022.35	127	1	1	94.46 ±9.04	1.73 ±0.47	1.82 ± 0.41	1046.63	159	3	96	96.74 ±7.03	8.14 ±0.81	8.42 ± 0.59	1027.55	105	2	80	95.50 ±8.39	7.19 ±0.85	7.52 ± 0.52
p_b^7	b	1005.32	3	20	1	61.97 ±21.39	25.00 ±8.59	40.30 ±1.35	1005.02	3	20	10	60.99 ±19.23	28.10 ±8.79	45.90 ±1.21	1005.01	4	19	8	64.20 ±21.81	27.30 ±9.10	42.40 ±1.22
	w	1022.35	127	1	1	94.46 ±9.04	1.72 ±0.45	1.78 ± 0.41	1022.35	127	1	1	94.46 ±9.04	1.72 ±0.47	1.81 ± 0.42	1022.35	127	1	1	94.46 ±9.04	1.77 ±0.42	1.84 ± 0.37
p_b^8	b	***1003.54**	5	20	74	70.32 ±19.70	58.30 ±16.50	83.00 ±1.27	1005.05	3	19	4	69.46 ±17.98	27.80 ±7.09	39.90 ±0.90	1005.14	3	20	4	66.59 ±18.73	28.70 ±8.01	43.20 ±1.28
	w	1022.35	127	1	1	94.46 ±9.04	1.65 ±0.48	1.75 ± 0.43	1108.25	249	10	100	98.73 ±3.72	25.90 ±1.26	26.20 ± 0.85	1032.65	112	9	90	90.28 ±15.11	36.60 ±6.14	40.50 ± 0.74
p_b^9	b	1005.32	3	20	1	61.97 ±21.39	25.00 ±8.62	40.30 ±1.11	1005.03	4	20	6	69.05 ±17.05	30.70 ±7.52	44.60 ±1.22	1005.10	4	20	7	69.34 ±16.95	32.30 ±7.95	46.50 ±1.16
	w	1022.35	127	1	1	94.46 ±9.04	1.68 ±0.47	1.78 ± 0.41	1086.51	232	7	100	98.79 ±4.17	19.60 ±1.02	19.90 ± 0.68	1029.12	124	3	28	93.84 ±10.09	7.92 ±0.96	8.42 ± 0.53

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.3: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra150_16* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	1008.95	5	20	1	66.74 ±17.77	33.20 ±8.49	49.70 ±1.82	**1008.49**	4	20	10	76.87 ±14.37	42.00 ±8.09	54.70 ±1.48	**1008.42**	5	20	95	64.07 ±18.88	57.40 ±17.00	89.90 ±1.57
	w	1106.23	315	1	1	99.60 ±1.57	1.92 ±0.27	1.94 ± 0.24	1106.23	315	1	1	99.60 ±1.57	1.92 ±0.27	1.92 ± 0.27	1106.23	315	1	1	99.60 ±1.57	1.95 ±0.26	1.96 ± 0.24
p_b^1	b	1007.31	10	19	75	76.41 ±16.50	68.50 ±14.90	89.70 ±1.65	1008.77	4	20	2	71.86 ±18.16	36.30 ±9.08	50.60 ±1.41	1008.56	4	20	3	69.77 ±18.37	36.30 ±9.50	52.10 ±1.14
	w	1106.23	315	1	1	99.60 ±1.57	1.84 ±0.39	1.84 ± 0.39	1218.04	381	6	90	100.00 ±0.00	17.30 ±0.67	17.30 ± 0.67	1144.36	292	3	79	99.97 ±0.30	12.20 ±0.66	12.20 ± 0.66
p_b^2	b	1006.29	19	20	74	70.00 ±20.26	67.50 ±19.30	96.90 ±1.38	1008.87	5	20	2	71.41 ±16.49	35.90 ±8.51	50.30 ±1.65	1008.64	4	20	3	71.63 ±17.35	37.20 ±9.30	52.00 ±1.24
	w	1106.23	315	1	1	99.60 ±1.57	1.90 ±0.36	1.90 ± 0.36	1258.64	435	16	91	100.00 ±0.00	51.00 ±1.32	51.00 ± 1.32	1162.89	314	7	98	99.85 ±1.40	33.60 ±1.01	33.60 ± 0.90
p_b^3	b	1008.91	4	20	3	65.61 ±15.56	33.60 ±8.04	51.30 ±1.29	1008.85	5	20	15	72.02 ±18.68	40.70 ±10.50	56.50 ±1.37	1008.79	5	20	4	68.58 ±18.61	36.00 ±9.74	52.40 ±1.17
	w	1106.23	315	1	1	99.60 ±1.57	1.90 ±0.30	1.91 ± 0.29	1127.81	277	2	100	99.67 ±1.36	6.35 ±0.55	6.37 ± 0.54	1106.23	315	1	1	99.60 ±1.57	1.93 ±0.32	1.94 ± 0.31
p_b^4	b	1008.95	5	20	1	66.74 ±17.77	33.60 ±9.02	50.40 ±1.30	1008.60	4	20	6	76.04 ±13.58	40.60 ±7.31	53.30 ±1.49	1008.51	5	20	11	73.51 ±14.24	41.60 ±8.21	56.60 ±1.35
	w	1106.23	315	1	1	99.60 ±1.57	1.98 ±0.24	1.98 ± 0.24	1143.58	337	2	92	99.97 ±0.30	5.94 ±0.42	5.94 ± 0.42	1109.61	277	2	91	99.37 ±2.38	7.46 ±0.62	7.53 ± 0.56
p_b^5	b	1007.65	12	20	81	75.71 ±19.55	76.80 ±19.80	101.00 ±1.89	1008.76	4	20	2	71.07 ±17.39	36.30 ±8.91	51.10 ±1.45	1008.64	4	20	4	72.64 ±17.96	38.40 ±9.73	52.70 ±1.37
	w	1106.23	315	1	1	99.60 ±1.57	1.93 ±0.29	1.94 ± 0.28	1217.30	380	16	97	100.00 ±0.00	55.30 ±1.38	55.30 ± 1.38	1145.43	369	7	92	99.68 ±2.61	33.50 ±1.28	33.60 ± 0.96
p_b^6	b	1008.95	5	20	1	66.74 ±17.77	33.50 ±8.86	50.20 ±1.35	1008.69	5	20	3	77.04 ±15.63	40.00 ±8.14	52.00 ±1.68	1008.55	5	20	10	75.96 ±15.65	43.50 ±9.14	57.30 ±1.19
	w	1106.23	315	1	1	99.60 ±1.57	1.95 ±0.33	1.96 ± 0.31	1153.28	341	2	93	99.56 ±2.44	6.09 ±0.55	6.12 ± 0.53	1127.40	255	3	95	99.76 ±1.63	13.30 ±0.70	13.30 ± 0.68
p_b^7	b	1008.83	5	20	82	65.69 ±19.51	60.80 ±18.10	92.70 ±1.96	1008.52	3	20	8	73.44 ±16.47	40.50 ±9.17	55.00 ±1.23	1008.55	4	20	70	66.95 ±19.73	58.90 ±17.50	88.00 ±1.79
	w	1106.23	315	1	1	99.60 ±1.57	1.89 ±0.34	1.92 ± 0.31	1106.23	315	1	1	99.60 ±1.57	1.93 ±0.29	1.94 ± 0.28	1106.23	315	1	1	99.60 ±1.57	1.94 ±0.28	1.96 ± 0.24
p_b^8	b	***1005.69**	7	20	76	66.36 ±19.68	66.20 ±19.60	100.00 ±1.64	1008.75	4	20	2	73.15 ±15.11	37.00 ±7.79	50.50 ±1.69	1008.62	4	19	3	73.18 ±16.58	35.70 ±8.23	48.80 ±1.19
	w	1106.23	315	1	1	99.60 ±1.57	1.95 ±0.30	1.95 ± 0.30	1255.52	398	10	95	99.99 ±0.10	30.00 ±0.89	30.00 ± 0.90	1166.23	293	10	97	99.87 ±0.70	52.70 ±1.15	52.80 ± 1.14
p_b^9	b	1007.49	11	18	100	75.48 ±17.63	77.30 ±18.20	102.00 ±1.17	1008.75	4	19	3	75.34 ±16.46	37.00 ±7.92	49.10 ±1.21	1008.60	6	20	6	76.41 ±14.45	42.30 ±8.06	55.40 ±1.31
	w	1106.23	315	1	1	99.60 ±1.57	1.96 ±0.31	1.97 ± 0.30	1219.71	427	13	95	99.98 ±0.20	44.60 ±1.04	44.70 ± 1.05	1142.26	345	17	85	99.60 ±1.67	97.40 ±2.21	97.80 ± 1.77

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.4: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra200_12* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	1204.92	2	20	1	55.33 ±23.40	34.50 ±14.80	62.50 ±1.71	1204.43	3	20	64	57.33 ±18.64	49.30 ±16.00	86.20 ±1.44	**1204.51**	3	20	39	57.94 ±20.90	45.60 ±16.60	78.80 ±1.78
	w	1212.94	50	1	1	89.09 ±12.29	2.30 ±0.61	2.56 ± 0.52	1212.94	50	1	1	89.09 ±12.29	2.41 ±0.55	2.67 ± 0.49	1212.94	50	1	1	89.09 ±12.29	2.35 ±0.57	2.69 ± 0.46
p_b^1	b	1203.11	5	20	100	55.77 ±21.59	64.90 ±25.10	117.00 ±1.84	1204.44	3	20	7	69.88 ±16.22	45.70 ±11.00	65.50 ±1.74	1204.63	3	20	3	57.94 ±21.92	37.30 ±14.20	64.30 ±1.76
	w	1212.94	50	1	1	89.09 ±12.29	2.36 ±0.57	2.64 ± 0.50	1240.38	141	3	72	98.67 ±3.68	9.82 ±0.61	9.95 ± 0.50	1231.56	123	7	99	96.31 ±6.36	35.60 ±2.54	37.00 ± 0.78
p_b^2	b	***1202.75**	4	19	91	56.05 ±20.59	59.80 ±22.30	107.00 ±1.63	1204.58	4	18	4	69.43 ±16.79	39.30 ±9.65	56.60 ±1.45	1204.61	3	20	5	63.12 ±20.08	41.10 ±13.20	65.20 ±1.77
	w	1212.94	50	1	1	89.09 ±12.29	2.41 ±0.55	2.70 ± 0.46	1286.11	167	20	66	98.51 ±6.64	74.10 ±5.22	75.20 ± 1.57	1236.97	128	8	90	96.79 ±6.81	40.60 ±2.96	41.90 ± 0.91
p_b^3	b	1204.70	3	20	2	56.53 ±23.64	35.70 ±15.10	63.20 ±1.68	1204.63	3	20	39	64.01 ±20.39	49.40 ±15.90	77.20 ±1.72	1204.65	3	19	2	54.79 ±22.89	32.80 ±13.90	59.50 ±1.71
	w	1212.94	50	1	1	89.09 ±12.29	2.37 ±0.63	2.71 ± 0.52	1215.77	97	2	83	91.65 ±10.92	6.61 ±0.89	7.24 ± 0.51	1212.94	50	1	1	89.09 ±12.29	2.33 ±0.58	2.61 ± 0.51
p_b^4	b	1204.92	2	20	1	55.33 ±23.40	34.70 ±14.80	62.90 ±1.61	**1204.32**	3	20	22	70.82 ±15.79	50.50 ±11.30	71.40 ±1.70	**1204.51**	3	20	7	60.09 ±24.46	40.20 ±16.40	66.90 ±1.77
	w	1212.94	50	1	1	89.09 ±12.29	2.32 ±0.60	2.62 ± 0.51	1223.39	89	3	63	95.70 ±9.14	9.48 ±1.02	9.86 ± 0.47	1214.48	117	2	70	91.30 ±9.97	7.16 ±0.91	7.83 ± 0.55
p_b^5	b	1203.16	5	20	93	62.19 ±20.69	74.10 ±24.80	119.00 ±1.95	1204.50	3	20	5	58.39 ±18.36	38.20 ±12.20	65.60 ±1.81	1204.66	3	20	3	58.97 ±21.74	37.90 ±14.00	64.30 ±2.00
	w	1212.94	50	1	1	89.09 ±12.29	2.40 ±0.58	2.71 ± 0.47	1241.04	168	4	72	98.54 ±5.00	13.10 ±0.94	13.30 ± 0.65	1230.37	116	11	91	96.55 ±6.07	58.50 ±3.92	60.50 ± 1.01
p_b^6	b	1204.92	2	20	1	55.33 ±23.40	34.60 ±14.80	62.40 ±1.57	1204.39	3	19	10	65.01 ±17.73	41.40 ±11.40	63.80 ±1.70	1204.57	3	20	8	61.05 ±19.70	41.60 ±13.50	68.30 ±1.95
	w	1212.94	50	1	1	89.09 ±12.29	2.34 ±0.64	2.63 ± 0.56	1225.90	112	3	82	95.35 ±7.93	10.20 ±0.95	10.60 ± 0.55	1217.90	105	2	90	92.29 ±9.98	9.09 ±1.11	9.82 ± 0.50
p_b^7	b	1204.90	5	20	94	55.07 ±23.49	59.20 ±25.20	108.00 ±1.91	1204.45	3	20	8	63.06 ±21.22	42.00 ±14.00	66.60 ±1.93	1204.53	3	19	6	58.07 ±21.31	36.10 ±13.30	62.10 ±1.73
	w	1212.94	50	1	1	89.09 ±12.29	2.40 ±0.60	2.63 ± 0.52	1212.94	50	1	1	89.09 ±12.29	2.28 ±0.58	2.55 ± 0.55	1212.94	50	1	1	89.09 ±12.29	2.33 ±0.57	2.61 ± 0.51
p_b^8	b	1202.84	4	20	87	54.88 ±22.54	63.50 ±26.10	116.00 ±1.79	1204.53	3	20	4	68.36 ±17.63	43.80 ±11.40	64.20 ±1.65	1204.57	3	20	3	61.32 ±22.03	39.50 ±14.20	64.40 ±1.81
	w	1212.94	50	1	1	89.09 ±12.29	2.34 ±0.65	2.65 ± 0.57	1256.35	166	6	100	98.28 ±4.56	21.00 ±1.22	21.30 ± 0.80	1239.36	121	17	91	97.69 ±6.52	102.00 ±6.97	104.00 ± 1.26
p_b^9	b	1203.02	4	20	99	50.97 ±20.49	70.20 ±28.50	138.00 ±1.86	1204.49	3	20	6	67.50 ±20.06	45.10 ±13.40	66.70 ±1.60	1204.71	3	19	2	56.64 ±23.08	33.90 ±13.90	60.10 ±1.50
	w	1212.94	50	1	1	89.09 ±12.29	2.40 ±0.58	2.67 ± 0.47	1240.49	119	3	100	98.00 ±5.65	11.00 ±0.86	11.20 ± 0.54	1229.92	107	10	95	95.86 ±8.09	61.40 ±5.32	64.10 ± 0.96

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.5: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra200_16* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	1208.34	4	20	1	70.30 ±16.84	50.60 ±12.20	72.30 ±2.96	**1207.91**	4	19	67	71.85 ±17.55	68.80 ±16.90	95.90 ±2.05	***1207.86**	5	19	67	70.69 ±17.39	68.80 ±17.10	97.50 ±2.05
	w	1335.63	342	1	1	99.74 ±1.38	2.85 ±0.43	2.85 ± 0.43	1335.63	342	1	1	99.74 ±1.38	2.81 ±0.39	2.82 ± 0.38	1335.63	342	1	1	99.74 ±1.38	2.84 ±0.42	2.85 ± 0.41
p_b^1	b	1208.34	4	20	1	70.30 ±16.84	51.20 ±12.30	73.00 ±2.31	1208.18	4	20	3	78.05 ±16.10	58.90 ±12.40	75.70 ±1.96	1208.09	4	20	6	80.08 ±14.82	62.20 ±11.80	77.80 ±2.09
	w	1335.63	342	1	1	99.74 ±1.38	2.84 ±0.42	2.84 ± 0.42	1377.57	329	2	84	100.00 ±0.00	7.52 ±0.56	7.52 ± 0.56	1335.63	342	1	1	99.74 ±1.38	2.85 ±0.41	2.86 ± 0.40
p_b^2	b	1208.34	4	20	1	70.30 ±16.84	52.00 ±12.60	74.00 ±2.07	1208.34	4	20	1	70.30 ±16.84	51.60 ±12.60	73.40 ±2.84	1207.99	5	20	4	78.96 ±14.76	60.90 ±11.80	77.00 ±1.90
	w	1335.63	342	1	1	99.74 ±1.38	2.84 ±0.37	2.84 ± 0.37	1411.49	423	10	96	99.60 ±2.01	41.00 ±1.37	41.10 ± 1.15	1335.63	342	1	1	99.74 ±1.38	2.85 ±0.38	2.85 ± 0.38
p_b^3	b	**1208.26**	5	20	2	72.71 ±16.33	54.00 ±12.30	74.30 ±2.02	1208.25	5	20	3	70.63 ±16.99	52.50 ±12.60	74.50 ±2.28	1208.17	4	20	33	73.15 ±17.24	65.00 ±15.40	88.80 ±2.21
	w	1335.63	342	1	1	99.74 ±1.38	2.80 ±0.53	2.80 ± 0.53	1343.70	295	2	84	99.94 ±0.51	7.98 ±0.49	7.98 ± 0.49	1335.63	342	1	1	99.74 ±1.38	2.83 ±0.40	2.84 ± 0.39
p_b^4	b	1208.34	4	20	1	70.30 ±16.84	51.70 ±12.50	73.70 ±1.98	1208.18	4	19	5	78.14 ±14.12	56.30 ±10.40	72.20 ±2.00	1208.10	4	20	12	75.94 ±15.58	61.20 ±13.00	80.70 ±1.92
	w	1335.63	342	1	1	99.74 ±1.38	2.81 ±0.42	2.82 ± 0.41	1338.04	291	2	88	99.85 ±1.20	7.62 ±0.61	7.63 ± 0.61	1335.63	342	1	1	99.74 ±1.38	2.89 ±0.42	2.89 ± 0.42
p_b^5	b	1208.34	4	20	1	70.30 ±16.84	52.30 ±12.70	74.40 ±1.91	1208.17	4	20	4	77.35 ±12.59	59.30 ±9.99	76.70 ±1.96	1208.15	4	20	4	72.54 ±15.91	55.30 ±12.30	76.20 ±2.10
	w	1335.63	342	1	1	99.74 ±1.38	2.85 ±0.46	2.86 ± 0.45	1372.66	390	2	83	99.90 ±0.99	7.54 ±0.56	7.55 ± 0.55	1335.63	342	1	1	99.74 ±1.38	2.76 ±0.47	2.77 ± 0.47
p_b^6	b	1208.34	4	20	1	70.30 ±16.84	52.20 ±12.60	74.40 ±2.07	1208.10	4	20	6	79.66 ±11.98	62.00 ±9.41	77.90 ±1.70	1208.02	4	20	7	77.89 ±16.04	61.60 ±13.00	79.10 ±2.26
	w	1335.63	342	1	1	99.74 ±1.38	2.92 ±0.34	2.93 ± 0.32	1347.83	297	2	88	99.63 ±2.08	7.82 ±0.46	7.85 ± 0.41	1335.63	342	1	1	99.74 ±1.38	2.88 ±0.38	2.88 ± 0.38
p_b^7	b	1208.34	4	20	1	70.30 ±16.84	52.10 ±12.80	74.30 ±1.84	1207.96	5	19	10	74.52 ±15.31	55.90 ±11.80	75.10 ±2.06	1207.94	4	20	12	74.18 ±15.75	60.50 ±13.00	81.50 ±1.87
	w	1335.63	342	1	1	99.74 ±1.38	2.79 ±0.45	2.79 ± 0.45	1335.63	342	1	1	99.74 ±1.38	2.79 ±0.41	2.79 ± 0.41	1335.63	342	1	1	99.74 ±1.38	2.85 ±0.38	2.85 ± 0.38
p_b^8	b	1208.34	4	20	1	70.30 ±16.84	52.40 ±12.50	74.70 ±1.93	1208.33	3	20	2	75.96 ±14.78	56.40 ±11.10	74.50 ±2.40	1208.12	4	19	2	74.31 ±16.09	52.80 ±11.50	71.20 ±2.14
	w	1335.63	342	1	1	99.74 ±1.38	2.88 ±0.35	2.88 ± 0.35	1396.84	416	5	89	99.80 ±1.53	18.90 ±0.81	18.90 ± 0.79	1335.63	342	1	1	99.74 ±1.38	2.78 ±0.46	2.80 ± 0.45
p_b^9	b	1208.34	4	20	1	70.30 ±16.84	52.00 ±12.20	74.50 ±1.97	1208.26	4	19	2	74.34 ±17.81	52.70 ±12.80	70.80 ±1.97	1208.19	4	19	3	71.58 ±17.28	51.40 ±12.70	71.80 ±2.03
	w	1335.63	342	1	1	99.74 ±1.38	2.80 ±0.40	2.81 ± 0.39	1374.00	343	2	84	99.85 ±1.49	7.74 ±0.52	7.75 ± 0.52	1335.63	342	1	1	99.74 ±1.38	2.83 ±0.38	2.83 ± 0.38

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.6: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra250_12* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	1404.16	2	20	1	47.68 ±22.28	41.10 ±19.10	86.10 ±2.70	1403.67	3	19	42	59.11 ±19.93	57.30 ±19.60	96.80 ±2.67	**1403.91**	2	20	12	55.32 ±22.13	50.90 ±20.60	91.90 ±3.01
	w	1408.33	62	1	1	86.46 ±10.81	3.15 ±0.70	3.62 ± 0.56	1408.33	62	1	1	86.46 ±10.81	3.08 ±0.63	3.57 ± 0.53	1408.33	62	1	1	86.46 ±10.81	3.18 ±0.71	3.67 ± 0.47
p_b^1	b	1404.16	2	20	1	47.68 ±22.28	41.20 ±19.20	86.60 ±2.73	1403.76	2	19	8	68.10 ±17.96	57.90 ±15.30	85.30 ±2.42	1403.96	2	20	5	57.66 ±21.95	52.00 ±20.20	90.20 ±3.36
	w	1408.33	62	1	1	86.46 ±10.81	3.11 ±0.66	3.63 ± 0.59	1421.90	82	4	88	95.02 ±9.03	16.90 ±1.77	17.80 ± 0.63	1410.74	65	8	89	90.15 ±12.45	45.80 ±6.30	50.70 ± 0.99
p_b^2	b	1404.16	2	20	1	47.68 ±22.28	41.20 ±19.00	86.40 ±2.52	***1403.65**	3	19	41	78.89 ±13.68	71.80 ±12.90	91.10 ±2.39	1403.97	2	20	2	55.81 ±22.50	48.80 ±19.70	87.20 ±3.23
	w	1408.33	62	1	1	86.46 ±10.81	3.17 ±0.57	3.63 ± 0.52	1447.84	104	18	86	97.54 ±9.03	90.00 ±8.67	92.20 ± 2.42	1411.72	64	13	97	90.29 ±11.94	84.20 ±11.30	93.20 ± 1.79
p_b^3	b	**1404.07**	2	20	2	53.60 ±24.75	46.10 ±21.50	86.10 ±2.93	1403.96	3	20	33	58.57 ±22.01	58.20 ±22.20	99.20 ±2.72	1403.98	2	20	11	51.71 ±19.91	47.00 ±18.30	90.60 ±2.81
	w	1408.33	62	1	1	86.46 ±10.81	3.14 ±0.55	3.58 ± 0.51	1408.61	37	2	92	88.23 ±12.37	8.21 ±1.21	9.31 ± 0.52	1408.33	62	1	1	86.46 ±10.81	3.10 ±0.54	3.57 ± 0.49
p_b^4	b	1404.16	2	20	1	47.68 ±22.28	41.40 ±19.30	87.10 ±2.64	1403.68	3	20	12	56.24 ±17.96	51.50 ±17.00	91.50 ±3.38	1403.92	2	20	8	60.23 ±21.54	54.80 ±19.80	91.20 ±2.86
	w	1408.33	62	1	1	86.46 ±10.81	3.19 ±0.61	3.66 ± 0.47	1410.49	75	3	93	91.33 ±10.79	12.50 ±1.56	13.80 ± 0.54	1409.62	58	2	93	86.96 ±12.31	9.26 ±1.42	10.60 ± 0.59
p_b^5	b	1404.16	2	20	1	47.68 ±22.28	40.90 ±19.10	86.00 ±2.93	1403.77	4	20	41	79.41 ±14.24	79.20 ±14.50	99.80 ±2.48	1404.03	3	18	6	56.19 ±20.40	45.60 ±16.70	81.10 ±2.26
	w	1408.33	62	1	1	86.46 ±10.81	3.07 ±0.67	3.66 ± 0.51	1421.53	94	5	85	92.47 ±11.67	20.80 ±2.60	22.50 ± 0.66	1410.95	75	3	95	90.67 ±10.22	17.90 ±2.04	19.60 ± 0.64
p_b^6	b	1404.16	2	20	1	47.68 ±22.28	41.30 ±19.30	86.80 ±2.67	***1403.65**	2	19	10	62.69 ±17.85	54.30 ±15.70	86.60 ±2.54	1403.96	2	20	6	56.55 ±22.16	51.50 ±20.40	91.00 ±3.01
	w	1408.33	62	1	1	86.46 ±10.81	3.12 ±0.59	3.63 ± 0.50	1413.47	89	3	94	93.01 ±9.45	13.00 ±1.55	13.90 ± 0.62	1409.32	39	2	57	88.18 ±11.33	9.10 ±1.25	10.30 ± 0.52
p_b^7	b	1404.16	2	20	1	47.68 ±22.28	41.60 ±19.70	87.20 ±2.55	1403.80	3	20	8	54.05 ±21.14	48.90 ±19.20	90.70 ±2.70	**1403.91**	3	20	8	49.54 ±22.42	45.80 ±21.10	92.50 ±2.67
	w	1408.33	62	1	1	86.46 ±10.81	3.04 ±0.63	3.55 ± 0.61	1408.33	62	1	1	86.46 ±10.81	3.07 ±0.60	3.59 ± 0.60	1408.33	62	1	1	86.46 ±10.81	3.11 ±0.61	3.59 ± 0.53
p_b^8	b	1404.16	2	20	1	47.68 ±22.28	41.20 ±19.10	86.30 ±2.53	1403.71	4	19	40	79.96 ±13.07	72.80 ±12.20	91.30 ±2.51	1403.99	2	18	4	60.36 ±19.74	47.80 ±15.90	79.30 ±2.56
	w	1408.33	62	1	1	86.46 ±10.81	3.15 ±0.62	3.61 ± 0.53	1436.91	141	19	85	95.97 ±9.01	95.40 ±9.24	99.50 ± 2.13	1411.71	71	12	92	89.91 ±12.97	80.20 ±11.60	89.30 ± 1.49
p_b^9	b	1404.16	2	20	1	47.68 ±22.28	41.30 ±19.60	86.60 ±2.53	1403.78	3	20	9	68.01 ±15.70	62.80 ±14.60	92.30 ±2.81	1404.00	2	20	7	60.38 ±20.88	55.90 ±19.50	92.90 ±3.05
	w	1408.33	62	1	1	86.46 ±10.81	3.13 ±0.67	3.60 ± 0.55	1423.25	96	4	89	96.33 ±6.03	17.80 ±1.42	18.50 ± 0.71	1411.03	51	8	99	91.38 ±11.53	57.10 ±7.33	62.50 ± 0.98

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.7: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra250_16* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	1406.29	4	19	1	66.41 ±16.82	62.20 ±16.20	93.70 ±3.29	**1405.83**	3	20	72	67.05 ±17.78	87.10 ±23.00	130.00 ±3.68	**1405.90**	3	19	84	68.69 ±20.76	88.90 ±27.00	130.00 ±3.16
	w	1499.30	247	1	1	99.16 ±3.83	3.88 ±0.33	3.92 ± 0.27	1499.30	247	1	1	99.16 ±3.83	3.88 ±0.33	3.90 ± 0.30	1499.30	247	1	1	99.16 ±3.83	3.88 ±0.43	3.92 ± 0.39
p_b^1	b	1404.78	6	20	85	62.69 ±19.94	97.90 ±31.00	156.00 ±3.76	1405.99	3	20	3	70.31 ±17.17	71.50 ±17.80	102.00 ±3.20	1406.03	4	20	5	71.79 ±14.86	75.00 ±16.00	104.00 ±3.18
	w	1499.30	247	1	1	99.16 ±3.83	3.73 ±0.49	3.78 ± 0.46	1500.90	249	2	94	99.56 ±2.68	9.73 ±0.60	9.77 ± 0.55	1499.30	247	1	1	99.16 ±3.83	3.88 ±0.33	3.89 ± 0.31
p_b^2	b	1404.29	5	19	83	61.83 ±20.35	93.10 ±30.70	151.00 ±2.72	1405.95	3	20	3	76.24 ±14.91	78.20 ±15.40	103.00 ±2.38	1406.13	4	19	4	74.08 ±17.24	72.10 ±16.80	97.60 ±2.83
	w	1499.30	247	1	1	99.16 ±3.83	3.79 ±0.52	3.82 ± 0.48	1506.28	263	2	96	99.51 ±2.09	9.84 ±0.58	9.89 ± 0.56	1499.30	247	1	1	99.16 ±3.83	3.82 ±0.50	3.86 ± 0.45
p_b^3	b	1406.29	4	19	1	66.41 ±16.82	62.90 ±16.70	94.70 ±2.85	1406.14	4	20	2	63.61 ±17.14	64.60 ±17.40	101.00 ±3.07	1406.08	3	20	79	66.16 ±18.16	89.90 ±25.00	136.00 ±3.20
	w	1499.30	247	1	1	99.16 ±3.83	3.80 ±0.47	3.84 ± 0.44	1499.30	247	1	1	99.16 ±3.83	3.87 ±0.44	3.90 ± 0.41	1499.30	247	1	1	99.16 ±3.83	3.90 ±0.39	3.91 ± 0.38
p_b^4	b	1406.29	4	19	1	66.41 ±16.82	62.80 ±15.90	94.70 ±3.22	1405.93	2	20	9	74.43 ±14.68	78.40 ±16.00	106.00 ±3.02	1405.96	4	19	77	74.32 ±16.69	97.90 ±22.30	131.00 ±2.82
	w	1499.30	247	1	1	99.16 ±3.83	3.73 ±0.49	3.78 ± 0.46	1499.30	247	1	1	99.16 ±3.83	3.86 ±0.42	3.89 ± 0.37	1499.30	247	1	1	99.16 ±3.83	3.80 ±0.47	3.85 ± 0.38
p_b^5	b	1404.94	6	20	95	64.46 ±20.04	108.00 ±33.70	168.00 ±3.47	1406.07	4	20	5	75.07 ±14.67	78.00 ±15.70	104.00 ±2.82	1406.12	3	20	5	69.11 ±17.40	72.10 ±18.30	104.00 ±3.03
	w	1499.30	247	1	1	99.16 ±3.83	3.93 ±0.35	3.95 ± 0.33	1499.30	247	1	1	99.16 ±3.83	3.82 ±0.43	3.85 ± 0.41	1499.30	247	1	1	99.16 ±3.83	3.86 ±0.38	3.90 ± 0.33
p_b^6	b	1406.29	4	19	1	66.41 ±16.82	63.60 ±16.40	95.70 ±2.95	1406.03	2	20	4	72.79 ±15.49	75.40 ±16.30	104.00 ±3.11	1405.95	3	20	7	69.91 ±18.03	73.80 ±19.40	106.00 ±3.20
	w	1499.30	247	1	1	99.16 ±3.83	3.85 ±0.43	3.87 ± 0.42	1499.30	247	1	1	99.16 ±3.83	3.85 ±0.43	3.89 ± 0.37	1499.30	247	1	1	99.16 ±3.83	3.86 ±0.38	3.90 ± 0.33
p_b^7	b	1406.29	4	19	1	66.41 ±16.82	63.20 ±15.90	95.10 ±3.01	1405.97	3	20	41	67.30 ±18.76	82.40 ±23.20	123.00 ±2.99	1405.92	3	20	70	65.51 ±21.11	91.70 ±29.60	140.00 ±3.83
	w	1499.30	247	1	1	99.16 ±3.83	3.84 ±0.42	3.86 ± 0.38	1499.30	247	1	1	99.16 ±3.83	3.93 ±0.29	3.95 ± 0.26	1499.30	247	1	1	99.16 ±3.83	3.89 ±0.40	3.91 ± 0.38
p_b^8	b	***1404.19**	5	20	90	59.03 ±20.45	97.00 ±33.40	164.00 ±4.17	1406.07	4	20	3	76.37 ±14.09	78.90 ±14.20	103.00 ±2.49	1406.16	4	20	2	68.67 ±18.67	70.50 ±19.50	103.00 ±3.03
	w	1499.30	247	1	1	99.16 ±3.83	3.84 ±0.48	3.87 ± 0.46	1500.53	305	4	97	99.47 ±2.71	20.00 ±0.90	20.10 ± 0.72	1499.30	247	1	1	99.16 ±3.83	3.81 ±0.48	3.86 ± 0.40
p_b^9	b	1404.99	7	20	88	65.22 ±19.16	116.00 ±34.00	177.00 ±3.34	1406.09	4	20	4	73.16 ±15.98	75.30 ±16.40	103.00 ±3.40	1406.10	4	20	6	73.02 ±17.27	77.60 ±18.40	106.00 ±3.02
	w	1499.30	247	1	1	99.16 ±3.83	3.85 ±0.43	3.90 ± 0.36	1499.30	247	1	1	99.16 ±3.83	3.94 ±0.28	3.97 ± 0.22	1499.30	247	1	1	99.16 ±3.83	3.90 ±0.36	3.91 ± 0.35

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.8: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra300_12* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	1603.67	3	20	1	51.68 ±21.91	57.60 ±24.70	111.00 ±3.33	1603.45	2	20	8	48.55 ±23.49	55.50 ±26.90	115.00 ±3.50	**1603.52**	2	20	5	52.58 ±22.94	59.70 ±26.30	114.00 ±4.67
	w	1607.53	13	1	1	86.32 ±11.58	4.09 ±0.67	4.76 ± 0.49	1607.53	13	1	1	86.32 ±11.58	4.09 ±0.71	4.75 ± 0.43	1607.53	13	1	1	86.32 ±11.58	4.15 ±0.71	4.79 ± 0.50
p_b^1	b	1603.57	4	19	96	50.92 ±23.79	81.70 ±38.40	160.00 ±3.30	1603.32	3	19	60	74.62 ±15.26	91.00 ±18.70	122.00 ±2.84	1603.64	3	20	7	58.21 ±22.46	68.80 ±26.90	118.00 ±4.22
	w	1607.91	6	18	14	46.54 ±23.58	50.90 ±25.80	109.00 ± 3.05	1614.51	65	3	89	90.53 ±12.08	15.00 ±2.09	16.60 ± 0.63	1609.13	48	2	48	86.29 ±12.88	10.40 ±1.55	12.00 ± 0.58
p_b^2	b	**1603.51**	4	19	85	51.88 ±24.14	83.00 ±39.20	160.00 ±3.19	1603.17	2	19	59	70.35 ±14.90	83.40 ±17.60	119.00 ±2.90	1603.59	3	20	5	55.21 ±20.16	63.80 ±23.50	115.00 ±4.39
	w	1607.97	7	18	13	50.01 ±25.75	54.30 ±27.90	109.00 ± 3.12	1631.79	102	14	92	90.12 ±17.08	79.60 ±15.40	88.30 ± 2.40	1609.59	59	3	44	85.38 ±14.87	15.80 ±2.82	18.50 ± 0.64
p_b^3	b	1603.67	3	20	1	51.68 ±21.91	57.40 ±24.40	111.00 ±3.38	1603.62	3	20	34	56.15 ±21.97	70.30 ±27.90	125.00 ±3.60	1603.65	3	20	17	54.19 ±23.18	64.10 ±27.70	118.00 ±4.09
	w	1607.53	13	1	1	86.32 ±11.58	4.05 ±0.78	4.71 ± 0.52	1608.10	76	2	93	85.62 ±11.95	9.88 ±1.47	11.60 ± 0.62	1607.53	13	1	1	86.32 ±11.58	4.02 ±0.80	4.67 ± 0.49
p_b^4	b	1603.67	3	20	1	51.68 ±21.91	56.90 ±24.60	110.00 ±3.57	1603.36	2	20	14	59.71 ±18.08	69.90 ±21.50	117.00 ±3.55	1603.58	2	19	7	52.96 ±20.71	57.70 ±22.70	109.00 ±3.09
	w	1607.53	13	1	1	86.32 ±11.58	4.18 ±0.78	4.75 ± 0.50	1609.38	63	2	91	89.26 ±11.33	10.10 ±1.31	11.20 ± 0.54	1607.74	62	2	24	81.37 ±16.14	8.55 ±1.75	10.40 ± 0.64
p_b^5	b	1603.53	4	20	91	53.45 ±24.43	93.30 ±42.60	174.00 ±3.66	1603.32	3	20	56	74.99 ±14.75	96.20 ±19.00	128.00 ±2.76	1603.64	3	20	5	53.53 ±22.08	61.80 ±25.70	116.00 ±4.59
	w	1607.83	7	20	19	45.72 ±24.86	57.90 ±31.60	126.00 ± 3.48	1614.24	93	2	93	88.75 ±12.28	9.98 ±1.46	11.30 ± 0.51	1609.47	66	2	100	86.62 ±14.31	13.40 ±2.30	15.40 ± 0.56
p_b^6	b	1603.67	3	20	1	51.68 ±21.91	57.40 ±24.60	111.00 ±3.89	1603.38	3	20	99	74.41 ±17.46	106.00 ±25.00	143.00 ±3.45	1603.56	3	20	7	56.13 ±22.45	65.80 ±26.20	117.00 ±4.22
	w	1607.53	13	1	1	86.32 ±11.58	4.11 ±0.69	4.76 ± 0.49	1609.83	64	2	83	88.07 ±11.05	9.97 ±1.30	11.30 ± 0.58	1608.29	66	2	62	82.38 ±14.93	10.60 ±2.00	12.80 ± 0.54
p_b^7	b	1603.67	3	20	1	51.68 ±21.91	57.60 ±24.60	111.00 ±3.66	1603.47	2	20	4	51.21 ±20.21	58.00 ±23.00	113.00 ±3.12	1603.57	3	20	24	52.61 ±23.05	67.10 ±29.70	128.00 ±3.17
	w	1607.53	13	1	1	86.32 ±11.58	4.08 ±0.73	4.68 ± 0.53	1607.53	13	1	1	86.32 ±11.58	4.12 ±0.75	4.71 ± 0.50	1607.53	13	1	1	86.32 ±11.58	4.11 ±0.80	4.75 ± 0.48
p_b^8	b	1603.67	3	20	1	51.68 ±21.91	57.50 ±24.30	111.00 ±3.65	***1603.15**	2	20	40	74.59 ±14.36	90.70 ±17.80	122.00 ±2.90	1603.58	2	20	5	58.66 ±19.10	69.00 ±22.60	118.00 ±5.05
	w	1607.53	13	1	1	86.32 ±11.58	4.15 ±0.70	4.83 ± 0.38	1622.05	91	5	76	90.22 ±14.17	24.80 ±4.02	27.40 ± 0.80	1609.66	68	2	52	89.96 ±10.78	11.50 ±1.51	12.80 ± 0.56
p_b^9	b	1603.61	4	20	94	51.16 ±25.04	97.30 ±47.60	191.00 ±2.92	1603.33	3	20	75	77.44 ±14.94	104.00 ±20.50	135.00 ±3.01	1603.60	3	20	2	51.92 ±22.51	58.50 ±25.70	113.00 ±3.57
	w	1608.00	8	20	22	51.91 ±22.49	71.00 ±30.80	137.00 ± 2.77	1614.02	61	3	100	90.83 ±11.58	16.00 ±2.05	17.50 ± 0.67	1609.37	52	2	25	88.11 ±11.96	10.30 ±1.49	11.60 ± 0.58

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.9: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra300_16* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	1606.15	2	20	1	59.46 ±17.84	75.70 ±22.40	127.00 ±3.97	**1605.81**	3	19	90	70.95 ±18.07	111.00 ±28.30	157.00 ±4.14	1605.90	3	20	85	62.04 ±19.81	104.00 ±34.00	168.00 ±3.81
	w	1664.35	179	1	1	99.27 ±2.48	4.98 ±0.37	4.99 ± 0.36	1664.35	179	1	1	99.27 ±2.48	5.05 ±0.52	5.08 ± 0.48	1664.35	179	1	1	99.27 ±2.48	5.03 ±0.41	5.07 ± 0.35
p_b^1	b	1605.35	7	19	92	65.20 ±21.14	119.00 ±38.40	183.00 ±3.80	1605.96	3	20	5	73.71 ±13.56	96.90 ±18.30	131.00 ±3.16	1605.99	2	19	5	70.63 ±18.10	88.10 ±22.80	125.00 ±3.92
	w	1664.35	179	1	1	99.27 ±2.48	5.02 ±0.42	5.07 ± 0.41	1685.19	209	2	98	99.80 ±0.99	12.20 ±0.70	12.20 ± 0.67	1668.24	186	2	63	98.69 ±5.21	13.80 ±0.89	14.00 ± 0.50
p_b^2	b	***1605.06**	5	19	91	61.91 ±17.98	114.00 ±33.50	185.00 ±3.71	1606.03	2	20	2	63.97 ±17.50	83.20 ±23.40	130.00 ±4.10	1606.05	3	20	3	67.28 ±17.37	87.50 ±22.80	130.00 ±4.32
	w	1664.35	179	1	1	99.27 ±2.48	4.99 ±0.48	5.05 ± 0.41	1710.52	213	12	67	98.74 ±3.96	81.90 ±3.98	83.00 ± 2.37	1673.36	162	2	37	99.73 ±1.14	12.80 ±0.60	12.80 ± 0.60
p_b^3	b	1606.03	3	20	2	60.76 ±18.39	79.00 ±24.10	130.00 ±3.12	1606.05	3	20	8	63.92 ±18.55	84.20 ±24.30	132.00 ±4.05	1606.04	3	19	2	62.91 ±18.17	76.90 ±22.00	123.00 ±3.79
	w	1664.35	179	1	1	99.27 ±2.48	5.03 ±0.41	5.06 ± 0.40	1664.35	179	1	1	99.27 ±2.48	5.04 ±0.53	5.08 ± 0.48	1664.35	179	1	1	99.27 ±2.48	5.07 ±0.47	5.11 ± 0.42
p_b^4	b	1606.15	2	20	1	59.46 ±17.84	75.50 ±22.60	127.00 ±4.07	1605.90	3	20	6	69.70 ±15.61	92.10 ±20.60	132.00 ±4.06	1605.94	2	20	3	66.97 ±18.81	87.20 ±24.70	130.00 ±3.73
	w	1664.35	179	1	1	99.27 ±2.48	5.04 ±0.42	5.07 ± 0.38	1670.29	179	2	71	99.30 ±3.21	11.90 ±0.63	11.90 ± 0.52	1664.35	179	1	1	99.27 ±2.48	5.08 ±0.50	5.12 ± 0.47
p_b^5	b	1605.22	5	20	93	65.98 ±19.61	130.00 ±39.20	197.00 ±3.76	1605.98	3	19	4	70.29 ±15.21	87.70 ±19.10	125.00 ±3.37	1606.05	4	18	3	66.36 ±18.48	77.10 ±21.40	116.00 ±3.23
	w	1664.35	179	1	1	99.27 ±2.48	5.01 ±0.52	5.05 ± 0.46	1688.02	202	2	98	99.47 ±2.33	12.30 ±0.62	12.40 ± 0.58	1668.71	183	2	100	97.28 ±8.18	16.10 ±1.56	16.50 ± 0.76
p_b^6	b	1606.15	2	20	1	59.46 ±17.84	76.50 ±22.80	129.00 ±3.82	1605.88	3	20	4	69.36 ±15.49	92.00 ±20.60	133.00 ±3.17	1605.95	3	20	5	67.09 ±17.26	89.60 ±23.40	134.00 ±3.62
	w	1664.35	179	1	1	99.27 ±2.48	5.10 ±0.50	5.11 ± 0.49	1674.64	158	2	83	99.61 ±2.18	12.30 ±0.67	12.30 ± 0.67	1665.54	176	2	74	99.08 ±3.25	14.30 ±0.70	14.40 ± 0.56
p_b^7	b	1606.15	2	20	1	59.46 ±17.84	76.40 ±22.80	129.00 ±3.17	1605.84	3	19	7	67.39 ±17.36	84.30 ±21.30	125.00 ±3.98	**1605.86**	2	19	5	64.74 ±18.94	81.00 ±23.80	125.00 ±3.93
	w	1664.35	179	1	1	99.27 ±2.48	4.98 ±0.42	5.01 ± 0.41	1664.35	179	1	1	99.27 ±2.48	5.01 ±0.46	5.04 ± 0.47	1664.35	179	1	1	99.27 ±2.48	5.11 ±0.53	5.16 ± 0.48
p_b^8	b	1605.33	4	19	91	60.86 ±20.80	115.00 ±39.40	188.00 ±4.17	1606.01	2	18	2	69.52 ±17.59	80.90 ±20.40	116.00 ±3.69	1606.05	2	20	2	64.94 ±19.73	84.00 ±25.60	129.00 ±4.18
	w	1664.35	179	1	1	99.27 ±2.48	5.03 ±0.46	5.05 ± 0.43	1696.36	217	3	83	99.48 ±2.09	18.00 ±0.68	18.10 ± 0.63	1669.48	200	2	60	98.68 ±4.68	14.20 ±0.80	14.40 ± 0.54
p_b^9	b	1605.32	5	20	98	61.50 ±22.25	124.00 ±44.60	201.00 ±6.51	1605.91	2	20	5	71.51 ±15.18	94.50 ±19.90	132.00 ±4.03	1606.07	2	20	3	65.25 ±19.31	86.10 ±26.20	132.00 ±3.37
	w	1664.35	179	1	1	99.27 ±2.48	5.09 ±0.53	5.13 ± 0.52	1686.33	201	2	98	99.69 ±1.28	12.40 ±0.62	12.50 ± 0.59	1668.03	181	2	55	99.16 ±2.42	14.70 ±0.63	14.80 ± 0.53

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.10: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra350_12* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{T} \pm s$ $[10^{-3}]$
p_b^0	b	1807.24	4	20	1	63.67 ±22.61	88.00 ±31.20	139.00 ±4.66	1806.52	4	19	68	69.85 ±17.53	111.00 ±28.30	159.00 ±3.47	1806.77	4	20	40	65.70 ±20.59	106.00 ±33.30	160.00 ±3.54
	w	1826.67	87	1	1	95.63 ±7.13	5.72 ±0.62	6.00 ± 0.35	1826.67	87	1	1	95.63 ±7.13	5.80 ±0.60	6.05 ± 0.30	1826.67	87	1	1	95.63 ±7.13	5.72 ±0.68	5.99 ± 0.46
p_b^1	b	1807.24	4	20	1	63.67 ±22.61	88.00 ±31.30	138.00 ±4.53	1806.81	4	20	7	78.58 ±13.34	113.00 ±19.60	143.00 ±4.68	1807.07	5	20	2	60.58 ±21.59	84.30 ±30.40	140.00 ±4.29
	w	1826.67	87	1	1	95.63 ±7.13	5.71 ±0.59	6.01 ± 0.36	1883.57	207	3	95	99.57 ±1.84	20.70 ±0.74	20.80 ± 0.69	1870.10	156	13	100	99.25 ±3.28	131.00 ±4.68	132.00 ± 2.04
p_b^2	b	1807.24	4	20	1	63.67 ±22.61	88.60 ±31.90	139.00 ±4.39	1806.97	4	20	2	64.52 ±18.39	90.50 ±26.20	140.00 ±4.34	1807.08	4	20	3	69.57 ±18.19	99.00 ±26.50	142.00 ±4.38
	w	1826.67	87	1	1	95.63 ±7.13	5.78 ±0.72	6.01 ± 0.52	1955.88	263	17	80	99.94 ±0.60	133.00 ±3.16	133.00 ± 3.09	1877.09	218	15	76	99.33 ±2.09	146.00 ±4.05	147.00 ± 2.77
p_b^3	b	**1807.14**	4	20	2	64.72 ±21.38	90.00 ±29.80	139.00 ±4.10	1807.06	4	20	43	69.33 ±17.95	109.00 ±28.40	158.00 ±4.08	1807.14	4	20	4	61.33 ±21.32	86.50 ±30.40	141.00 ±4.58
	w	1826.67	87	1	1	95.63 ±7.13	5.84 ±0.72	6.11 ± 0.44	1832.24	166	2	98	96.63 ±6.06	13.70 ±1.16	14.20 ± 0.71	1826.67	87	1	1	95.63 ±7.13	5.79 ±0.59	6.03 ± 0.36
p_b^4	b	1807.24	4	20	1	63.67 ±22.61	87.50 ±31.10	138.00 ±4.18	1806.59	4	20	16	76.88 ±14.46	114.00 ±22.70	148.00 ±4.12	1806.94	4	20	8	66.63 ±18.60	98.20 ±27.90	147.00 ±5.30
	w	1826.67	87	1	1	95.63 ±7.13	5.74 ±0.59	6.04 ± 0.34	1853.77	111	3	93	98.75 ±4.63	20.80 ±1.13	21.10 ± 0.66	1832.88	147	2	98	96.01 ±7.01	15.00 ±1.27	15.60 ± 0.56
p_b^5	b	1807.24	4	20	1	63.67 ±22.61	88.00 ±31.40	138.00 ±4.17	1806.84	5	19	7	76.88 ±14.20	104.00 ±19.80	136.00 ±4.64	1807.14	4	20	5	64.99 ±19.53	94.30 ±28.10	145.00 ±4.72
	w	1826.67	87	1	1	95.63 ±7.13	5.74 ±0.66	6.00 ± 0.47	1886.41	169	4	99	99.55 ±2.36	28.00 ±1.06	28.10 ± 0.75	1870.17	153	18	100	99.56 ±2.10	193.00 ±4.76	194.00 ± 3.00
p_b^6	b	1807.24	4	20	1	63.67 ±22.61	87.50 ±31.50	137.00 ±5.10	1806.63	3	20	6	73.37 ±17.06	105.00 ±24.80	144.00 ±4.66	1806.89	5	20	7	66.59 ±19.84	97.40 ±29.60	146.00 ±5.09
	w	1826.67	87	1	1	95.63 ±7.13	5.74 ±0.58	5.98 ± 0.35	1861.50	160	5	76	99.62 ±1.77	35.70 ±1.06	35.80 ± 0.95	1842.89	123	6	95	97.95 ±5.00	51.70 ±2.77	52.80 ± 1.07
p_b^7	b	1807.24	4	20	1	63.67 ±22.61	88.70 ±31.60	139.00 ±5.30	***1806.50**	4	20	7	66.75 ±20.02	96.00 ±28.90	144.00 ±4.48	**1806.67**	4	20	7	61.10 ±21.08	89.80 ±31.70	147.00 ±5.08
	w	1826.67	87	1	1	95.63 ±7.13	5.70 ±0.62	5.94 ± 0.42	1826.67	87	1	1	95.63 ±7.13	5.69 ±0.64	6.00 ± 0.49	1826.67	87	1	1	95.63 ±7.13	5.80 ±0.63	6.03 ± 0.48
p_b^8	b	1807.24	4	20	1	63.67 ±22.61	89.20 ±32.10	140.00 ±4.58	1806.92	4	20	3	74.20 ±16.26	105.00 ±23.10	141.00 ±3.86	1807.14	4	19	4	73.30 ±17.98	98.90 ±24.30	135.00 ±4.58
	w	1826.67	87	1	1	95.63 ±7.13	5.75 ±0.57	6.02 ± 0.40	1900.07	246	7	100	99.27 ±3.53	50.10 ±2.01	50.50 ± 1.07	1875.67	192	12	63	99.25 ±2.73	112.00 ±3.73	113.00 ± 2.51
p_b^9	b	1807.24	4	20	1	63.67 ±22.61	88.30 ±31.80	139.00 ±4.74	1806.91	5	19	6	74.50 ±15.55	101.00 ±21.50	136.00 ±3.98	1807.10	4	20	2	67.78 ±21.30	95.20 ±30.70	140.00 ±3.69
	w	1826.67	87	1	1	95.63 ±7.13	5.69 ±0.66	5.96 ± 0.42	1886.54	225	6	95	99.60 ±1.79	43.60 ±1.31	43.80 ± 0.96	1868.01	156	18	90	99.51 ±1.87	200.00 ±5.26	201.00 ± 3.44

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.11: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra350_16* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	1805.84	3	20	1	61.18 ±18.78	97.80 ±30.30	160.00 ±4.25	***1805.53**	3	20	97	68.58 ±19.48	137.00 ±39.60	200.00 ±4.89	**1805.60**	4	20	88	65.55 ±18.87	130.00 ±37.30	199.00 ±5.11
	w	1850.14	135	1	1	99.14 ±2.83	6.41 ±0.60	6.46 ± 0.59	1850.14	135	1	1	99.14 ±2.83	6.33 ±0.65	6.38 ± 0.63	1850.14	135	1	1	99.14 ±2.83	6.31 ±0.63	6.36 ± 0.59
p_b^1	b	1805.84	3	20	1	61.18 ±18.78	96.10 ±29.90	158.00 ±4.22	1805.62	3	20	4	72.93 ±15.58	118.00 ±25.70	162.00 ±4.66	1805.80	3	20	6	73.10 ±16.12	120.00 ±26.50	164.00 ±4.42
	w	1850.14	135	1	1	99.14 ±2.83	6.30 ±0.59	6.36 ± 0.57	1882.72	197	3	97	99.79 ±1.90	22.70 ±0.92	22.80 ± 0.79	1867.22	167	2	87	99.65 ±1.73	18.20 ±0.73	18.20 ± 0.70
p_b^2	b	1805.84	3	20	1	61.18 ±18.78	95.30 ±28.70	156.00 ±5.91	1805.76	3	18	2	68.90 ±17.57	98.30 ±25.60	142.00 ±4.30	1805.83	3	20	2	61.01 ±15.96	98.80 ±26.70	162.00 ±4.06
	w	1850.14	135	1	1	99.14 ±2.83	6.35 ±0.65	6.42 ± 0.59	1918.82	216	17	63	99.93 ±0.70	148.00 ±4.13	148.00 ± 4.04	1871.74	152	2	88	99.68 ±1.83	19.20 ±0.80	19.30 ± 0.72
p_b^3	b	**1805.72**	2	20	2	65.15 ±18.56	104.00 ±29.20	159.00 ±4.90	1805.71	3	20	22	65.15 ±15.51	110.00 ±26.70	168.00 ±5.30	1805.79	3	20	19	67.96 ±15.42	116.00 ±26.40	170.00 ±4.14
	w	1850.14	135	1	1	99.14 ±2.83	6.41 ±0.62	6.44 ± 0.60	1856.71	146	2	100	99.19 ±3.79	15.40 ±0.78	15.50 ± 0.57	1850.14	135	1	1	99.14 ±2.83	6.24 ±0.57	6.30 ± 0.56
p_b^4	b	1805.84	3	20	1	61.18 ±18.78	98.00 ±30.00	160.00 ±3.68	1805.63	3	20	5	71.51 ±16.51	115.00 ±27.40	161.00 ±4.63	1805.74	3	20	4	69.54 ±16.99	113.00 ±28.10	162.00 ±4.62
	w	1850.14	135	1	1	99.14 ±2.83	6.33 ±0.57	6.40 ± 0.53	1866.48	153	2	79	99.88 ±0.94	14.70 ±0.52	14.70 ± 0.52	1855.06	175	2	55	98.67 ±4.76	15.10 ±1.02	15.30 ± 0.69
p_b^5	b	1805.84	3	20	1	61.18 ±18.78	99.40 ±30.40	163.00 ±3.62	1805.65	3	18	2	68.43 ±17.03	99.60 ±24.80	145.00 ±3.37	1805.83	3	19	4	65.84 ±16.55	102.00 ±25.80	155.00 ±4.33
	w	1850.14	135	1	1	99.14 ±2.83	6.31 ±0.66	6.35 ± 0.61	1879.76	183	2	95	99.17 ±5.12	14.90 ±0.98	15.00 ± 0.65	1866.47	164	11	91	99.30 ±4.41	121.00 ±6.04	122.00 ± 2.79
p_b^6	b	1805.84	3	20	1	61.18 ±18.78	97.80 ±30.30	160.00 ±4.78	1805.67	3	19	4	71.89 ±14.12	111.00 ±22.30	154.00 ±4.74	1805.76	3	17	6	69.04 ±17.39	96.00 ±24.60	139.00 ±4.27
	w	1850.14	135	1	1	99.14 ±2.83	6.28 ±0.57	6.33 ± 0.57	1869.15	137	3	80	99.84 ±0.94	22.80 ±0.88	22.80 ± 0.87	1859.94	118	2	78	99.44 ±2.63	17.30 ±0.79	17.40 ± 0.68
p_b^7	b	1805.84	3	20	1	61.18 ±18.78	96.30 ±30.00	157.00 ±4.69	1805.56	3	20	61	63.99 ±19.54	124.00 ±38.50	192.00 ±5.28	1805.63	2	20	32	64.69 ±18.46	116.00 ±32.80	180.00 ±5.01
	w	1850.14	135	1	1	99.14 ±2.83	6.33 ±0.53	6.38 ± 0.51	1850.14	135	1	1	99.14 ±2.83	6.27 ±0.53	6.30 ± 0.52	1850.14	135	1	1	99.14 ±2.83	6.33 ±0.60	6.38 ± 0.60
p_b^8	b	1805.84	3	20	1	61.18 ±18.78	98.80 ±29.90	161.00 ±4.14	1805.77	3	20	2	69.35 ±16.82	111.00 ±26.50	161.00 ±4.30	1805.83	3	19	2	65.87 ±17.06	100.00 ±26.10	152.00 ±4.44
	w	1850.14	135	1	1	99.14 ±2.83	6.41 ±0.58	6.45 ± 0.57	1895.23	144	3	95	100.00 ±0.00	22.80 ±0.69	22.80 ± 0.69	1867.80	145	3	77	99.73 ±1.09	29.20 ±0.90	29.20 ± 0.80
p_b^9	b	1805.84	3	20	1	61.18 ±18.78	97.90 ±30.60	160.00 ±4.76	1805.61	3	20	3	68.12 ±18.25	110.00 ±29.50	161.00 ±4.71	1805.83	2	19	2	61.07 ±18.53	93.50 ±28.90	153.00 ±4.50
	w	1850.14	135	1	1	99.14 ±2.83	6.38 ±0.56	6.45 ± 0.52	1880.40	153	2	98	99.88 ±0.67	15.30 ±0.60	15.30 ± 0.59	1865.26	137	2	69	99.60 ±1.98	18.80 ±0.77	18.80 ± 0.67

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.12: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra400_12* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	**2004.28**	3	20	1	60.29 ±20.02	103.00 ±34.00	170.00 ±4.78	2003.95	2	20	42	65.21 ±18.13	123.00 ±34.30	189.00 ±4.66	2004.07	3	19	98	63.37 ±20.91	128.00 ±42.30	202.00 ±3.86
	w	2025.39	120	1	1	95.59 ±8.50	7.04 ±0.88	7.40 ± 0.63	2025.39	120	1	1	95.59 ±8.50	7.13 ±0.84	7.41 ± 0.57	2025.39	120	1	1	95.59 ±8.50	7.06 ±0.81	7.37 ± 0.64
p_b^1	b	**2004.28**	3	20	1	60.29 ±20.02	102.00 ±33.70	169.00 ±4.48	2004.05	2	19	7	73.98 ±15.93	121.00 ±26.70	164.00 ±5.58	2004.18	2	19	5	64.60 ±17.29	106.00 ±28.80	165.00 ±4.39
	w	2025.39	120	1	1	95.59 ±8.50	7.14 ±0.79	7.49 ± 0.54	2031.95	140	3	97	96.52 ±6.99	24.20 ±1.88	25.10 ± 0.69	2028.03	113	3	54	94.35 ±9.10	26.10 ±2.50	27.50 ± 0.70
p_b^2	b	**2004.28**	3	20	1	60.29 ±20.02	102.00 ±34.50	170.00 ±4.38	2004.14	2	20	4	70.60 ±17.18	121.00 ±29.70	171.00 ±5.27	2004.19	3	20	5	65.88 ±18.01	116.00 ±31.60	176.00 ±4.84
	w	2028.75	290	16	53	92.11 ±8.94	154.00 ±15.60	168.00 ± 3.57	2067.29	197	19	88	91.87 ±13.90	167.00 ±25.70	181.00 ± 4.01	2027.68	126	3	52	92.42 ±11.32	25.80 ±3.20	28.00 ± 0.81
p_b^3	b	**2004.28**	3	20	1	60.29 ±20.02	102.00 ±34.50	169.00 ±6.22	2004.23	3	20	11	63.08 ±22.45	109.00 ±38.90	173.00 ±4.07	2004.24	3	18	6	63.48 ±19.77	98.20 ±30.40	155.00 ±4.10
	w	2025.39	120	1	1	95.59 ±8.50	7.09 ±0.86	7.39 ± 0.69	2025.39	120	1	1	95.59 ±8.50	7.15 ±0.83	7.53 ± 0.56	2025.39	120	1	1	95.59 ±8.50	7.07 ±0.85	7.39 ± 0.58
p_b^4	b	**2004.28**	3	20	1	60.29 ±20.02	102.00 ±34.00	170.00 ±5.43	2003.91	3	20	10	66.49 ±17.78	118.00 ±32.80	176.00 ±5.52	2004.11	2	19	11	64.17 ±17.97	107.00 ±30.40	167.00 ±4.33
	w	2025.39	120	1	1	95.59 ±8.50	7.07 ±0.92	7.35 ± 0.62	2027.26	93	2	73	96.68 ±6.88	15.80 ±1.30	16.40 ± 0.67	2025.39	120	1	1	95.59 ±8.50	7.07 ±0.75	7.33 ± 0.55
p_b^5	b	**2004.28**	3	20	1	60.29 ±20.02	103.00 ±34.20	170.00 ±4.77	2004.02	2	19	8	72.64 ±15.98	119.00 ±26.40	164.00 ±5.09	2004.21	3	20	8	68.97 ±18.21	126.00 ±32.90	182.00 ±5.76
	w	2025.39	120	1	1	95.59 ±8.50	7.10 ±0.84	7.40 ± 0.57	2032.73	153	2	97	96.49 ±6.71	16.00 ±1.25	16.60 ± 0.63	2027.81	115	3	73	94.58 ±8.59	28.30 ±2.85	29.80 ± 0.86
p_b^6	b	**2004.28**	3	20	1	60.29 ±20.02	103.00 ±34.40	170.00 ±4.76	***2003.85**	3	20	12	74.06 ±16.46	132.00 ±29.60	179.00 ±4.40	2004.06	3	20	5	60.72 ±21.96	106.00 ±38.90	175.00 ±5.84
	w	2025.39	120	1	1	95.59 ±8.50	7.14 ±0.79	7.48 ± 0.59	2029.11	123	2	76	94.67 ±9.77	15.50 ±1.73	16.40 ± 0.72	2025.39	120	1	1	95.59 ±8.50	7.04 ±0.91	7.32 ± 0.60
p_b^7	b	**2004.28**	3	20	1	60.29 ±20.02	102.00 ±34.30	170.00 ±4.57	2004.04	2	20	10	60.51 ±20.22	107.00 ±36.10	176.00 ±5.19	**2004.05**	2	20	7	62.40 ±18.96	109.00 ±33.20	176.00 ±5.45
	w	2025.39	120	1	1	95.59 ±8.50	7.05 ±0.86	7.38 ± 0.63	2025.39	120	1	1	95.59 ±8.50	7.02 ±0.94	7.38 ± 0.63	2025.39	120	1	1	95.59 ±8.50	7.04 ±0.86	7.40 ± 0.60
p_b^8	b	**2004.28**	3	20	1	60.29 ±20.02	102.00 ±34.10	169.00 ±5.14	2004.05	3	20	5	74.42 ±15.44	129.00 ±27.00	173.00 ±5.14	2004.18	4	20	5	70.79 ±17.95	126.00 ±32.00	178.00 ±6.90
	w	2027.44	200	15	52	92.70 ±12.05	147.00 ±19.00	158.00 ± 3.42	2035.54	160	4	81	96.42 ±7.10	32.30 ±2.53	33.50 ± 0.84	2029.40	107	3	52	95.92 ±8.71	27.40 ±2.58	28.50 ± 0.74
p_b^9	b	**2004.28**	3	20	1	60.29 ±20.02	102.00 ±33.80	170.00 ±4.40	2004.08	3	20	7	71.11 ±15.89	125.00 ±28.50	176.00 ±5.61	2004.14	4	20	6	63.98 ±19.39	114.00 ±35.30	179.00 ±5.96
	w	2025.39	120	1	1	95.59 ±8.50	7.04 ±0.77	7.34 ± 0.57	2034.39	152	2	97	96.72 ±6.60	16.30 ±1.25	16.80 ± 0.61	2027.98	139	2	74	95.09 ±8.78	19.80 ±1.97	20.80 ± 0.70

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.13: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra400_16* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	2006.05	3	20	1	68.77 ±18.17	132.00 ±35.50	192.00 ±5.32	2005.77	3	20	38	72.57 ±15.25	153.00 ±32.30	212.00 ±4.41	***2005.70**	3	20	40	69.40 ±16.23	148.00 ±34.20	213.00 ±5.66
	w	2085.98	203	1	1	99.49 ±2.50	7.93 ±0.38	7.95 ± 0.36	2085.98	203	1	1	99.49 ±2.50	7.84 ±0.46	7.88 ± 0.41	2085.98	203	1	1	99.49 ±2.50	7.76 ±0.53	7.79 ± 0.50
p_b^1	b	2006.05	3	20	1	68.77 ±18.17	133.00 ±35.80	193.00 ±4.21	2005.94	2	20	2	69.63 ±16.85	135.00 ±33.60	194.00 ±5.17	2006.05	3	20	1	68.77 ±18.17	132.00 ±34.50	192.00 ±4.46
	w	2085.98	203	1	1	99.49 ±2.50	7.77 ±0.60	7.82 ± 0.56	2104.11	260	2	91	99.78 ±1.00	17.80 ±0.70	17.80 ± 0.68	2090.87	191	2	38	99.47 ±2.46	18.20 ±0.88	18.30 ± 0.66
p_b^2	b	2006.05	3	20	1	68.77 ±18.17	133.00 ±35.30	192.00 ±4.38	2006.05	3	20	1	68.77 ±18.17	132.00 ±35.30	192.00 ±5.36	2006.05	3	20	1	68.77 ±18.17	133.00 ±35.10	193.00 ±4.82
	w	2085.98	203	1	1	99.49 ±2.50	7.80 ±0.55	7.84 ± 0.48	2132.84	305	7	84	98.88 ±4.02	67.90 ±3.16	68.80 ± 1.75	2094.58	228	2	35	99.44 ±2.92	18.40 ±0.97	18.50 ± 0.67
p_b^3	b	2006.04	3	19	2	69.44 ±17.12	127.00 ±32.10	182.00 ±5.23	2005.97	3	20	2	70.10 ±18.65	135.00 ±35.80	192.00 ±5.78	2005.97	3	19	11	68.65 ±18.06	129.00 ±34.40	188.00 ±4.72
	w	2085.98	203	1	1	99.49 ±2.50	7.88 ±0.43	7.92 ± 0.42	2085.98	203	1	1	99.49 ±2.50	7.82 ±0.43	7.87 ± 0.39	2085.98	203	1	1	99.49 ±2.50	7.80 ±0.55	7.84 ± 0.44
p_b^4	b	2006.05	3	20	1	68.77 ±18.17	131.00 ±34.80	190.00 ±5.82	2005.87	3	19	9	78.97 ±12.97	148.00 ±24.80	187.00 ±4.14	2005.97	3	19	8	72.46 ±15.08	136.00 ±28.70	188.00 ±5.22
	w	2085.98	203	1	1	99.49 ±2.50	7.85 ±0.50	7.88 ± 0.45	2091.26	318	2	75	99.83 ±0.91	17.60 ±0.71	17.60 ± 0.71	2088.02	267	2	66	98.49 ±5.99	18.40 ±1.25	18.70 ± 0.63
p_b^5	b	2006.05	3	20	1	68.77 ±18.17	132.00 ±34.30	192.00 ±4.97	2005.98	3	20	5	77.29 ±14.52	153.00 ±28.90	198.00 ±4.80	2006.03	3	20	3	70.01 ±17.20	136.00 ±34.00	195.00 ±5.48
	w	2085.98	203	1	1	99.49 ±2.50	7.71 ±0.50	7.74 ± 0.46	2104.05	269	2	91	99.77 ±0.96	17.80 ±0.60	17.80 ± 0.58	2093.80	225	2	26	98.85 ±3.99	17.50 ±0.91	17.70 ± 0.54
p_b^6	b	2006.05	3	20	1	68.77 ±18.17	132.00 ±36.00	192.00 ±5.12	2005.92	3	19	4	75.58 ±14.28	141.00 ±27.60	186.00 ±5.54	2005.94	3	20	3	69.82 ±15.84	138.00 ±32.10	198.00 ±4.47
	w	2085.98	203	1	1	99.49 ±2.50	7.77 ±0.49	7.83 ± 0.43	2094.90	222	2	54	99.71 ±1.54	17.30 ±0.74	17.40 ± 0.66	2093.34	254	2	85	97.54 ±7.00	20.40 ±1.45	20.90 ± 0.58
p_b^7	b	2006.05	3	20	1	68.77 ±18.17	133.00 ±35.90	194.00 ±4.51	**2005.75**	3	20	12	67.78 ±16.96	136.00 ±34.00	201.00 ±5.09	2005.82	2	20	10	67.48 ±18.28	135.00 ±37.10	200.00 ±4.75
	w	2085.98	203	1	1	99.49 ±2.50	7.72 ±0.65	7.74 ± 0.63	2085.98	203	1	1	99.49 ±2.50	7.80 ±0.58	7.85 ± 0.52	2085.98	203	1	1	99.49 ±2.50	7.78 ±0.48	7.83 ± 0.43
p_b^8	b	**2005.78**	14	15	99	79.65 ±16.07	156.00 ±32.40	196.00 ±6.44	2006.01	3	20	2	71.08 ±16.06	138.00 ±31.80	194.00 ±6.03	2006.05	3	20	1	68.77 ±18.17	133.00 ±34.80	193.00 ±4.58
	w	2085.98	203	1	1	99.49 ±2.50	7.85 ±0.57	7.88 ± 0.53	2109.50	329	2	99	99.75 ±1.98	17.80 ±0.68	17.90 ± 0.63	2094.88	280	2	48	98.99 ±3.22	19.30 ±0.85	19.50 ± 0.59
p_b^9	b	2006.05	3	20	1	68.77 ±18.17	133.00 ±35.20	194.00 ±5.50	2005.89	3	20	4	73.54 ±14.47	144.00 ±28.90	196.00 ±5.13	2005.94	4	18	4	74.39 ±15.87	133.00 ±28.60	178.00 ±4.65
	w	2085.98	203	1	1	99.49 ±2.50	7.83 ±0.55	7.88 ± 0.49	2105.24	267	2	91	99.70 ±1.59	18.10 ±0.72	18.10 ± 0.67	2094.33	246	2	38	98.99 ±4.03	19.10 ±1.05	19.30 ± 0.62

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.14: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra450_12* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	2205.07	2	19	1	61.48 ±21.64	118.00 ±42.10	193.00 ±5.00	2204.56	3	19	10	68.35 ±20.53	135.00 ±40.90	197.00 ±5.10	**2204.69**	3	20	83	64.70 ±20.75	159.00 ±50.60	245.00 ±5.14
	w	2235.32	146	1	1	96.15 ±7.70	8.67 ±0.86	9.02 ± 0.42	2235.32	146	1	1	96.15 ±7.70	8.53 ±0.89	8.84 ± 0.54	2235.32	146	1	1	96.15 ±7.70	8.62 ±0.85	8.95 ± 0.46
p_b^1	b	2205.07	2	19	1	61.48 ±21.64	119.00 ±42.10	193.00 ±4.97	2204.64	3	20	5	69.11 ±16.63	143.00 ±34.70	208.00 ±6.36	2204.95	3	18	5	67.51 ±18.88	126.00 ±34.60	187.00 ±5.33
	w	2235.32	146	1	1	96.15 ±7.70	8.64 ±0.81	8.95 ± 0.46	2246.23	171	2	96	97.95 ±5.99	19.40 ±1.29	19.80 ± 0.54	2237.64	164	2	55	95.63 ±10.97	20.40 ±2.25	21.30 ± 0.64
p_b^2	b	2205.07	2	19	1	61.48 ±21.64	119.00 ±42.20	193.00 ±4.61	2204.84	3	19	3	71.34 ±17.81	139.00 ±34.90	195.00 ±5.06	2204.93	3	20	5	71.16 ±17.15	148.00 ±35.90	208.00 ±6.63
	w	2235.32	146	1	1	96.15 ±7.70	8.56 ±0.84	8.91 ± 0.58	2287.04	189	20	93	94.12 ±10.49	216.00 ±24.90	230.00 ± 5.20	2237.18	166	2	25	95.28 ±10.14	18.90 ±2.17	19.90 ± 0.46
p_b^3	b	**2205.02**	3	20	9	68.98 ±19.12	143.00 ±39.70	207.00 ±4.99	2204.92	2	20	12	64.06 ±20.01	132.00 ±41.30	206.00 ±4.97	2204.94	3	20	2	61.24 ±17.89	126.00 ±37.30	206.00 ±6.38
	w	2235.32	146	1	1	96.15 ±7.70	8.60 ±0.93	8.95 ± 0.54	2235.32	146	1	1	96.15 ±7.70	8.60 ±0.84	8.96 ± 0.42	2235.32	146	1	1	96.15 ±7.70	8.61 ±0.85	8.94 ± 0.42
p_b^4	b	2205.07	2	19	1	61.48 ±21.64	118.00 ±41.90	192.00 ±4.92	2204.50	3	20	9	66.86 ±16.50	138.00 ±34.10	208.00 ±6.03	2204.73	3	20	7	62.95 ±19.66	134.00 ±41.90	213.00 ±6.97
	w	2235.32	146	1	1	96.15 ±7.70	8.68 ±0.84	9.00 ± 0.42	2235.32	146	1	1	96.15 ±7.70	8.62 ±0.77	8.98 ± 0.40	2235.32	146	1	1	96.15 ±7.70	8.62 ±0.83	9.01 ± 0.46
p_b^5	b	2205.07	2	19	1	61.48 ±21.64	118.00 ±42.30	192.00 ±4.97	2204.67	3	20	7	72.66 ±16.34	151.00 ±33.70	208.00 ±6.78	2204.91	4	20	4	62.53 ±19.75	129.00 ±42.00	207.00 ±7.92
	w	2235.32	146	1	1	96.15 ±7.70	8.60 ±0.91	8.96 ± 0.42	2244.45	165	2	99	97.18 ±6.34	19.20 ±1.45	19.80 ± 0.63	2235.32	146	1	1	96.15 ±7.70	8.59 ±0.86	8.95 ± 0.41
p_b^6	b	2205.07	2	19	1	61.48 ±21.64	118.00 ±40.80	192.00 ±4.69	***2204.48**	3	20	7	71.50 ±19.01	149.00 ±39.30	209.00 ±7.01	2204.81	3	20	9	66.57 ±18.97	142.00 ±40.60	214.00 ±5.77
	w	2235.32	146	1	1	96.15 ±7.70	8.59 ±0.80	8.94 ± 0.42	2236.65	132	2	94	97.58 ±6.02	19.40 ±1.38	19.90 ± 0.68	2235.32	146	1	1	96.15 ±7.70	8.54 ±0.91	8.91 ± 0.45
p_b^7	b	2205.07	2	19	1	61.48 ±21.64	119.00 ±42.10	193.00 ±4.53	2204.61	3	20	5	68.32 ±19.72	142.00 ±40.70	208.00 ±5.68	2204.74	3	19	6	63.84 ±19.77	125.00 ±39.20	196.00 ±5.02
	w	2235.32	146	1	1	96.15 ±7.70	8.63 ±0.77	8.95 ± 0.46	2235.32	146	1	1	96.15 ±7.70	8.63 ±0.86	8.94 ± 0.56	2235.32	146	1	1	96.15 ±7.70	8.63 ±0.93	8.99 ± 0.59
p_b^8	b	2205.07	2	19	1	61.48 ±21.64	118.00 ±41.90	193.00 ±4.50	2204.77	3	19	4	74.91 ±14.24	146.00 ±28.70	195.00 ±5.20	2204.94	3	20	4	67.97 ±17.92	141.00 ±38.20	207.00 ±5.96
	w	2235.32	146	1	1	96.15 ±7.70	8.61 ±0.77	8.93 ± 0.47	2249.49	208	7	96	95.75 ±7.68	70.90 ±5.99	74.00 ± 1.68	2236.50	142	3	26	97.11 ±5.71	29.80 ±1.85	30.60 ± 0.90
p_b^9	b	2205.07	2	19	1	61.48 ±21.64	119.00 ±42.80	193.00 ±5.40	2204.71	3	20	5	69.46 ±16.54	144.00 ±34.20	208.00 ±5.61	2204.92	3	19	5	64.88 ±18.44	129.00 ±36.20	199.00 ±5.10
	w	2235.32	146	1	1	96.15 ±7.70	8.71 ±0.80	9.03 ± 0.39	2245.05	202	3	98	98.04 ±4.25	29.90 ±1.51	30.50 ± 0.75	2235.41	136	3	54	95.90 ±9.39	33.30 ±3.35	34.70 ± 0.84

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.15: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra450_16* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	2207.80	3	20	1	77.42 ±14.88	177.00 ±33.80	229.00 ±7.44	**2207.23**	4	19	97	78.67 ±12.97	204.00 ±34.60	259.00 ±6.02	***2207.18**	4	20	92	74.33 ±15.13	200.00 ±41.40	271.00 ±7.94
	w	2349.58	327	1	1	99.69 ±1.47	9.35 ±0.64	9.39 ± 0.61	2349.58	327	1	1	99.69 ±1.47	9.39 ±0.58	9.42 ± 0.57	2349.58	327	1	1	99.69 ±1.47	9.40 ±0.58	9.42 ± 0.57
p_b^1	b	2207.80	3	20	1	77.42 ±14.88	179.00 ±34.70	232.00 ±5.09	2207.71	4	18	3	81.45 ±12.58	168.00 ±25.70	206.00 ±5.42	2207.64	3	20	3	79.05 ±13.54	185.00 ±31.80	234.00 ±4.95
	w	2349.58	327	1	1	99.69 ±1.47	9.43 ±0.62	9.45 ± 0.61	2349.58	327	1	1	99.69 ±1.47	9.40 ±0.63	9.43 ± 0.62	2349.58	327	1	1	99.69 ±1.47	9.36 ±0.62	9.39 ± 0.61
p_b^2	b	2207.80	3	20	1	77.42 ±14.88	178.00 ±34.30	230.00 ±5.31	2207.68	4	18	2	81.60 ±11.86	169.00 ±24.90	207.00 ±5.48	2207.74	5	20	3	82.26 ±14.49	192.00 ±35.30	234.00 ±6.11
	w	2349.58	327	1	1	99.69 ±1.47	9.44 ±0.60	9.47 ± 0.61	2352.14	373	2	83	99.83 ±1.02	21.00 ±0.67	21.00 ± 0.62	2349.58	327	1	1	99.69 ±1.47	9.41 ±0.63	9.43 ± 0.62
p_b^3	b	**2207.76**	3	20	2	74.75 ±15.80	173.00 ±37.40	230.00 ±6.86	2207.61	4	20	12	79.88 ±12.60	189.00 ±30.70	237.00 ±5.52	2207.50	5	20	8	76.22 ±16.53	178.00 ±38.80	234.00 ±6.06
	w	2349.58	327	1	1	99.69 ±1.47	9.30 ±0.66	9.34 ± 0.64	2349.58	327	1	1	99.69 ±1.47	9.36 ±0.59	9.39 ± 0.58	2349.58	327	1	1	99.69 ±1.47	9.47 ±0.64	9.50 ± 0.62
p_b^4	b	2207.80	3	20	1	77.42 ±14.88	177.00 ±34.60	229.00 ±6.49	2207.56	4	20	3	78.77 ±14.14	184.00 ±33.60	233.00 ±6.30	2207.39	5	20	9	80.30 ±13.49	192.00 ±32.10	239.00 ±5.71
	w	2349.58	327	1	1	99.69 ±1.47	9.34 ±0.51	9.37 ± 0.50	2349.58	327	1	1	99.69 ±1.47	9.36 ±0.62	9.39 ± 0.63	2349.58	327	1	1	99.69 ±1.47	9.44 ±0.67	9.47 ± 0.66
p_b^5	b	2207.80	3	20	1	77.42 ±14.88	180.00 ±33.90	232.00 ±5.61	2207.66	4	19	3	82.44 ±11.56	179.00 ±25.70	216.00 ±6.54	2207.73	4	19	2	76.87 ±14.29	170.00 ±32.00	221.00 ±5.16
	w	2349.58	327	1	1	99.69 ±1.47	9.38 ±0.58	9.41 ± 0.55	2349.58	327	1	1	99.69 ±1.47	9.52 ±0.64	9.55 ± 0.65	2349.58	327	1	1	99.69 ±1.47	9.57 ±0.53	9.59 ± 0.53
p_b^6	b	2207.80	3	20	1	77.42 ±14.88	178.00 ±34.10	230.00 ±6.44	2207.49	3	20	3	81.72 ±12.96	189.00 ±30.00	232.00 ±6.72	2207.51	5	20	10	82.72 ±12.47	200.00 ±30.80	242.00 ±5.82
	w	2349.58	327	1	1	99.69 ±1.47	9.44 ±0.64	9.46 ± 0.61	2349.58	327	1	1	99.69 ±1.47	9.47 ±0.52	9.50 ± 0.50	2349.58	327	1	1	99.69 ±1.47	9.45 ±0.65	9.46 ± 0.64
p_b^7	b	2207.80	3	20	1	77.42 ±14.88	179.00 ±35.90	230.00 ±6.77	2207.24	4	20	9	77.50 ±14.27	184.00 ±34.10	238.00 ±5.10	***2207.18**	4	20	14	77.39 ±14.40	188.00 ±36.10	243.00 ±6.25
	w	2349.58	327	1	1	99.69 ±1.47	9.41 ±0.69	9.43 ± 0.68	2349.58	327	1	1	99.69 ±1.47	9.41 ±0.60	9.44 ± 0.59	2349.58	327	1	1	99.69 ±1.47	9.30 ±0.66	9.32 ± 0.65
p_b^8	b	2207.80	3	20	1	77.42 ±14.88	178.00 ±34.80	229.00 ±7.08	2207.80	3	20	1	77.42 ±14.88	179.00 ±35.90	231.00 ±7.15	2207.68	4	20	2	78.10 ±13.89	181.00 ±33.20	232.00 ±6.07
	w	2349.58	327	1	1	99.69 ±1.47	9.36 ±0.61	9.40 ± 0.62	2349.58	327	1	1	99.69 ±1.47	9.46 ±0.62	9.48 ± 0.61	2349.58	327	1	1	99.69 ±1.47	9.43 ±0.53	9.45 ± 0.55
p_b^9	b	2207.80	3	20	1	77.42 ±14.88	180.00 ±35.10	232.00 ±5.08	2207.72	4	20	2	76.63 ±14.47	176.00 ±33.40	230.00 ±6.30	2207.66	4	20	4	78.95 ±13.66	185.00 ±32.70	235.00 ±7.19
	w	2349.58	327	1	1	99.69 ±1.47	9.43 ±0.71	9.45 ± 0.68	2349.58	327	1	1	99.69 ±1.47	9.44 ±0.64	9.47 ± 0.62	2349.58	327	1	1	99.69 ±1.47	9.31 ±0.61	9.35 ± 0.61

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.16: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra500_12* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	2403.81	2	20	1	56.51 ±22.54	134.00 ±54.10	238.00 ±6.32	2403.49	3	19	34	58.43 ±20.49	141.00 ±49.00	241.00 ±6.55	2403.63	2	20	14	58.72 ±22.47	146.00 ±56.00	247.00 ±7.07
	w	2412.72	57	1	1	89.11 ±12.13	9.51 ±1.35	10.60 ± 0.57	2412.72	57	1	1	89.11 ±12.13	9.52 ±1.40	10.60 ± 0.61	2412.72	57	1	1	89.11 ±12.13	9.55 ±1.38	10.70 ± 0.60
p_b^1	b	2403.81	2	20	1	56.51 ±22.54	135.00 ±53.90	239.00 ±6.27	2403.48	3	20	80	81.80 ±12.31	216.00 ±33.20	264.00 ±5.47	2403.78	3	19	2	53.81 ±19.14	122.00 ±43.60	227.00 ±5.69
	w	2412.72	57	1	1	89.11 ±12.13	9.45 ±1.41	10.60 ± 0.59	2417.65	118	2	54	94.54 ±7.75	21.20 ±1.79	22.30 ± 0.66	2420.90	125	12	86	95.07 ±7.92	173.00 ±14.70	182.00 ± 4.48
p_b^2	b	2403.81	2	20	1	56.51 ±22.54	133.00 ±53.50	236.00 ±5.57	**2403.36**	4	19	63	81.97 ±11.94	203.00 ±30.60	247.00 ±5.28	2403.77	3	19	5	63.68 ±20.07	148.00 ±46.60	232.00 ±6.26
	w	2412.72	57	1	1	89.11 ±12.13	9.40 ±1.37	10.60 ± 0.60	2419.09	89	2	45	94.11 ±8.88	20.90 ±2.06	22.10 ± 0.70	2424.32	91	16	96	95.34 ±9.40	249.00 ±25.10	262.00 ± 4.04
p_b^3	b	2403.77	3	18	2	57.39 ±23.82	124.00 ±51.70	215.00 ±5.63	2403.73	3	20	41	60.42 ±20.43	157.00 ±54.00	259.00 ±4.46	2403.80	2	20	22	61.98 ±17.72	157.00 ±45.50	253.00 ±6.56
	w	2412.72	57	1	1	89.11 ±12.13	9.53 ±1.45	10.60 ± 0.56	2412.72	57	1	1	89.11 ±12.13	9.45 ±1.36	10.60 ± 0.59	2412.72	57	1	1	89.11 ±12.13	9.37 ±1.45	10.60 ± 0.57
p_b^4	b	2403.81	2	20	1	56.51 ±22.54	134.00 ±53.40	236.00 ±6.49	2403.47	3	20	13	65.40 ±19.96	161.00 ±49.60	246.00 ±5.86	2403.67	3	18	9	61.21 ±20.43	134.00 ±43.90	220.00 ±5.97
	w	2412.72	57	1	1	89.11 ±12.13	9.49 ±1.29	10.70 ± 0.62	2415.64	92	2	95	93.26 ±10.23	21.50 ±2.45	23.00 ± 0.69	2413.55	74	2	54	90.69 ±10.60	21.40 ±2.56	23.50 ± 0.71
p_b^5	b	2403.81	2	20	1	56.51 ±22.54	134.00 ±53.60	237.00 ±6.46	2403.50	3	20	70	78.51 ±13.79	207.00 ±36.90	263.00 ±5.05	2403.72	3	20	3	58.88 ±22.74	145.00 ±55.80	245.00 ±9.38
	w	2412.72	57	1	1	89.11 ±12.13	9.52 ±1.31	10.60 ± 0.53	2418.08	93	2	44	93.80 ±8.68	21.00 ±2.05	22.40 ± 0.67	2422.78	106	10	100	93.66 ±9.92	149.00 ±16.00	159.00 ± 3.01
p_b^6	b	2403.81	2	20	1	56.51 ±22.54	135.00 ±54.10	238.00 ±6.43	2403.48	3	20	7	59.45 ±18.58	144.00 ±44.40	243.00 ±5.40	2403.73	3	19	8	62.68 ±19.73	146.00 ±45.60	233.00 ±5.28
	w	2412.72	57	1	1	89.11 ±12.13	9.38 ±1.44	10.50 ± 0.56	2416.46	76	2	43	92.65 ±9.66	20.90 ±2.27	22.50 ± 0.70	2415.46	91	2	99	93.52 ±7.84	25.00 ±2.15	26.80 ± 0.72
p_b^7	b	2403.81	2	20	1	56.51 ±22.54	134.00 ±53.10	237.00 ±5.75	2403.51	2	19	8	58.19 ±21.69	134.00 ±49.40	230.00 ±5.36	**2403.62**	3	20	9	55.88 ±21.11	138.00 ±52.80	247.00 ±7.54
	w	2412.72	57	1	1	89.11 ±12.13	9.48 ±1.35	10.50 ± 0.65	2412.72	57	1	1	89.11 ±12.13	9.38 ±1.42	10.50 ± 0.61	2412.72	57	1	1	89.11 ±12.13	9.46 ±1.44	10.60 ± 0.53
p_b^8	b	***2402.58**	4	20	96	73.42 ±17.92	239.00 ±58.00	326.00 ±5.42	2403.38	3	20	63	79.85 ±14.60	207.00 ±38.20	259.00 ±4.79	2403.81	2	20	1	56.51 ±22.54	135.00 ±54.30	238.00 ±5.96
	w	2412.72	57	1	1	89.11 ±12.13	9.53 ±1.36	10.70 ± 0.54	2416.90	123	2	54	93.57 ±8.59	20.90 ±2.04	22.40 ± 0.71	2426.49	133	10	99	94.92 ±9.92	158.00 ±16.70	166.00 ± 2.97
p_b^9	b	2403.81	2	20	1	56.51 ±22.54	134.00 ±54.00	238.00 ±5.97	2403.42	3	20	70	79.10 ±13.87	210.00 ±37.10	265.00 ±5.39	2403.81	2	20	1	56.51 ±22.54	135.00 ±53.90	238.00 ±7.66
	w	2412.72	57	1	1	89.11 ±12.13	9.52 ±1.31	10.70 ± 0.53	2418.15	93	2	44	93.53 ±8.33	21.00 ±2.02	22.50 ± 0.83	2421.36	134	4	96	94.26 ±9.11	58.70 ±5.75	62.20 ± 1.05

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.17: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra500_16* [Dav06b]. Stopping criterion, $N_{it} = 100$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\bar{T} \pm s$ $[10^{-3}]$
p_b^0	b	**2407.35**	4	20	1	72.81 ±15.76	191.00 ±41.70	263.00 ±6.83	***2407.03**	3	20	45	79.93 ±12.88	229.00 ±37.90	286.00 ±7.30	2407.14	2	20	11	75.76 ±15.04	206.00 ±41.70	272.00 ±5.80
	w	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.72	11.10 ± 0.65	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.52	11.10 ± 0.49	2515.92	211	1	1	99.71 ±2.59	11.20 ±0.58	11.20 ± 0.57
p_b^1	b	**2407.35**	4	20	1	72.81 ±15.76	194.00 ±42.30	266.00 ±6.58	2407.35	4	20	1	72.81 ±15.76	193.00 ±41.80	265.00 ±6.86	2407.34	4	20	2	76.34 ±16.01	205.00 ±42.00	268.00 ±6.05
	w	2515.92	211	1	1	99.71 ±2.59	11.20 ±0.55	11.20 ± 0.52	2540.04	300	2	92	100.00 ±0.00	24.70 ±0.79	24.70 ± 0.79	2544.26	246	4	99	99.99 ±0.10	59.90 ±1.35	59.90 ± 1.35
p_b^2	b	**2407.35**	4	20	1	72.81 ±15.76	193.00 ±42.30	265.00 ±7.13	2407.35	4	20	1	72.81 ±15.76	193.00 ±42.20	266.00 ±6.62	2407.35	4	20	1	72.81 ±15.76	195.00 ±42.90	268.00 ±6.20
	w	2515.92	211	1	1	99.71 ±2.59	11.00 ±0.61	11.10 ± 0.58	2622.07	383	20	88	100.00 ±0.00	287.00 ±6.86	287.00 ± 6.86	2548.77	323	9	79	99.98 ±0.14	148.00 ±3.49	148.00 ± 3.48
p_b^3	b	**2407.35**	4	20	1	72.81 ±15.76	192.00 ±42.20	264.00 ±7.16	2407.23	4	20	21	76.00 ±14.03	215.00 ±40.60	282.00 ±5.13	2407.32	3	20	2	73.11 ±16.82	195.00 ±45.40	266.00 ±6.79
	w	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.61	11.10 ± 0.53	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.66	11.20 ± 0.58	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.57	11.10 ± 0.53
p_b^4	b	**2407.35**	4	20	1	72.81 ±15.76	191.00 ±41.80	262.00 ±6.34	2407.30	4	20	3	74.32 ±16.09	200.00 ±44.00	269.00 ±7.27	2407.34	4	20	2	76.34 ±16.01	206.00 ±42.40	270.00 ±6.36
	w	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.63	11.20 ± 0.54	2531.88	232	2	96	100.00 ±0.00	24.90 ±0.87	24.90 ± 0.87	2523.50	262	2	96	99.91 ±0.66	26.90 ±0.73	26.90 ± 0.71
p_b^5	b	**2407.35**	4	20	1	72.81 ±15.76	195.00 ±42.80	268.00 ±6.86	2407.28	4	20	2	74.03 ±15.97	198.00 ±43.20	266.00 ±6.45	2407.35	4	20	1	72.81 ±15.76	192.00 ±42.20	264.00 ±6.93
	w	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.59	11.10 ± 0.50	2545.07	285	2	92	100.00 ±0.00	24.70 ±0.72	24.70 ± 0.72	2545.44	310	4	94	99.76 ±2.39	63.20 ±2.00	63.40 ± 1.39
p_b^6	b	**2407.35**	4	20	1	72.81 ±15.76	193.00 ±42.30	265.00 ±6.67	2407.30	4	19	3	78.22 ±12.34	201.00 ±30.50	256.00 ±6.06	2407.35	4	20	1	72.81 ±15.76	193.00 ±41.10	265.00 ±5.69
	w	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.66	11.20 ± 0.58	2535.08	252	2	96	100.00 ±0.00	25.10 ±0.81	25.10 ± 0.81	2537.80	279	2	92	99.83 ±1.29	28.30 ±0.85	28.30 ± 0.74
p_b^7	b	**2407.35**	4	20	1	72.81 ±15.76	193.00 ±41.40	265.00 ±7.19	***2407.03**	4	20	10	74.05 ±15.56	201.00 ±41.90	271.00 ±5.44	**2407.09**	3	20	11	72.27 ±15.10	200.00 ±41.50	277.00 ±6.21
	w	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.67	11.10 ± 0.58	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.60	11.10 ± 0.52	2515.92	211	1	1	99.71 ±2.59	11.20 ±0.60	11.20 ± 0.50
p_b^8	b	**2407.35**	4	20	1	72.81 ±15.76	194.00 ±42.00	265.00 ±6.26	2407.35	4	20	1	72.81 ±15.76	195.00 ±42.00	267.00 ±5.51	2407.35	4	20	1	72.81 ±15.76	193.00 ±40.90	265.00 ±7.66
	w	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.60	11.10 ± 0.52	2544.20	274	2	98	99.91 ±0.90	24.70 ±0.82	24.70 ± 0.81	2551.93	277	15	96	100.00 ±0.00	249.00 ±7.60	249.00 ± 7.60
p_b^9	b	**2407.35**	4	20	1	72.81 ±15.76	197.00 ±42.40	270.00 ±5.78	2407.34	3	19	2	74.28 ±16.23	188.00 ±41.00	253.00 ±5.87	2407.35	4	20	1	72.81 ±15.76	192.00 ±41.70	263.00 ±6.50
	w	2515.92	211	1	1	99.71 ±2.59	11.10 ±0.58	11.10 ± 0.55	2543.15	281	2	92	100.00 ±0.00	25.00 ±0.77	25.00 ± 0.77	2543.07	260	6	97	100.00 ±0.00	96.60 ±2.06	96.60 ± 2.06

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.18: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra150_12* [Dav06b]. Stopping criterion, $T = 0.15[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	1004.13	2	6	1	670.68 ±389.43	70.60 ±41.00	1429.48 ±12.10	1003.73	3	3	9	1337.27 ±669.08	77.20 ±38.70	2605.77 ±30.11	1003.86	2	7	6	553.78 ±299.51	73.70 ±39.90	1130.23 ±10.81
	w	1005.88	4	18	97	112.89 ±54.35	74.10 ±35.60	229.89 ± 4.04	1004.59	3	19	98	104.01 ±53.95	71.00 ±36.80	220.27 ± 4.01	1004.67	3	19	86	109.92 ±54.44	74.10 ±36.90	223.49 ± 3.66
p_b^1	b	1003.81	5	17	74	153.67 ±50.61	97.30 ±32.10	238.30 ±3.56	1003.32	2	9	50	437.93 ±174.23	91.10 ±36.40	722.47 ±6.81	1003.79	3	11	6	351.53 ±174.22	75.00 ±37.30	703.64 ±6.37
	w	1009.64	7	19	14	177.56 ±83.00	79.10 ±36.90	338.21 ± 7.86	1010.48	84	20	100	209.87 ±42.53	122.00 ±24.00	259.22 ± 5.48	1006.54	7	20	91	134.46 ±31.28	114.00 ±26.70	177.18 ± 2.87
p_b^2	b	1003.00	5	19	73	133.10 ±46.15	95.30 ±33.00	210.75 ±3.17	**1003.12**	2	10	37	426.31 ±176.32	91.00 ±37.50	705.12 ±6.75	**1003.71**	3	2	4	2114.02 ±1014.73	78.70 ±37.70	4041.40 ±43.41
	w	1010.23	8	19	14	199.90 ±89.43	89.30 ±40.10	337.15 ± 6.64	1016.83	86	20	90	245.29 ±37.86	133.00 ±20.40	278.36 ± 6.13	1007.58	128	18	100	144.64 ±30.97	116.00 ±24.80	187.65 ± 2.35
p_b^3	b	1004.01	2	7	2	679.10 ±311.04	84.70 ±38.70	1206.50 ±11.22	1004.03	2	4	3	978.32 ±536.56	70.90 ±38.90	2076.41 ±16.34	1004.06	3	8	4	492.62 ±267.88	72.50 ±39.50	1022.31 ±8.66
	w	1005.42	3	20	100	120.01 ±43.68	90.10 ±32.60	201.21 ± 2.85	1004.90	3	20	97	109.28 ±43.88	78.00 ±32.10	208.72 ± 3.41	1004.87	3	11	94	188.11 ±91.81	72.50 ±35.10	391.14 ± 4.13
p_b^4	b	1004.13	2	3	1	1340.95 ±778.84	70.50 ±40.90	2859.26 ±29.35	1003.54	3	5	25	692.04 ±336.43	74.40 ±36.10	1398.37 ±14.08	1003.81	3	8	8	474.46 ±256.30	75.00 ±40.50	952.72 ±8.14
	w	1009.81	6	20	98	119.99 ±46.02	93.20 ±35.80	194.25 ± 3.29	1004.42	2	1	99	2794.38 ±1606.20	68.70 ±39.80	6108.24 ± 67.65	1004.88	4	13	98	157.91 ±71.26	76.80 ±34.90	309.64 ± 3.44
p_b^5	b	1004.01	6	17	79	145.31 ±46.76	99.30 ±31.90	220.64 ±3.08	1003.32	2	7	37	532.25 ±227.80	82.70 ±35.50	968.29 ±8.46	1003.84	3	5	6	795.26 ±338.02	76.60 ±32.50	1560.38 ±13.66
	w	1009.60	7	20	20	178.63 ±77.55	92.50 ±40.30	291.62 ± 6.38	1011.02	152	20	99	204.83 ±43.83	122.00 ±25.70	252.59 ± 5.71	1007.47	90	18	99	133.39 ±32.98	113.00 ±28.00	178.18 ± 2.06
p_b^6	b	1004.14	2	3	1	1327.96 ±763.33	69.80 ±40.10	2862.88 ±28.71	1003.51	2	7	30	496.91 ±223.05	80.10 ±35.80	934.27 ±7.18	1003.80	3	6	4	615.11 ±356.78	69.80 ±40.70	1326.39 ±13.28
	w	1009.69	9	20	94	111.58 ±42.78	96.60 ±37.10	174.76 ± 2.62	1004.42	2	1	97	2794.38 ±1606.20	68.40 ±39.50	6136.27 ± 73.43	1005.52	5	20	95	111.25 ±35.47	96.60 ±30.70	173.81 ± 2.67
p_b^7	b	1004.14	2	2	1	1991.62 ±1145.04	70.00 ±40.30	4277.98 ±47.51	1003.75	3	4	4	976.56 ±546.78	72.50 ±40.70	2019.14 ±18.48	1003.86	2	5	6	767.72 ±407.94	73.90 ±39.40	1560.23 ±14.58
	w	1005.26	4	18	98	95.35 ±50.60	71.90 ±38.20	199.59 ± 2.91	1004.87	2	16	100	109.34 ±53.88	72.50 ±35.40	227.57 ± 3.41	1004.92	3	17	96	101.62 ±47.16	72.90 ±33.80	210.02 ± 3.05
p_b^8	b	***1002.38**	4	20	72	113.77 ±45.45	90.50 ±36.30	189.38 ±3.30	1003.21	2	9	31	436.76 ±201.99	83.40 ±38.80	788.07 ±7.53	1003.79	3	9	3	437.74 ±227.02	73.10 ±38.00	901.12 ±8.62
	w	1008.53	8	20	14	204.38 ±68.65	99.70 ±33.80	309.02 ± 7.85	1011.22	76	19	98	241.60 ±47.85	129.00 ±25.20	282.87 ± 6.34	1008.47	55	19	100	122.90 ±29.35	117.00 ±27.90	158.85 ± 2.43
p_b^9	b	1004.14	2	1	1	3981.40 ±2291.61	70.00 ±39.90	8544.42 ±118.97	1003.33	3	11	37	319.45 ±129.45	83.50 ±33.70	575.45 ±6.56	1003.88	3	7	4	581.52 ±295.48	77.50 ±39.40	1127.55 ±10.65
	w	1009.87	8	20	16	168.98 ±82.57	87.60 ±42.80	290.74 ± 5.78	1015.31	154	20	98	188.30 ±42.58	119.00 ±26.80	239.17 ± 5.12	1010.92	114	19	99	115.59 ±24.76	121.00 ±25.90	143.86 ± 1.79

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.19: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra150_16* [Dav06b]. Stopping criterion, $T = 0.15[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	1007.24	4	1	1	3920.41 ±2019.98	76.00 ±39.00	7750.36 ±80.48	1006.87	4	5	6	782.27 ±372.91	80.60 ±38.30	1459.49 ±13.69	1006.87	3	4	6	987.61 ±457.06	81.90 ±37.90	1813.23 ±17.92
	w	1009.43	4	20	97	100.91 ±45.91	85.40 ±38.80	178.23 ± 4.47	1007.99	3	18	98	110.57 ±44.93	82.40 ±33.70	201.85 ± 4.28	1008.04	4	18	78	118.30 ±49.13	83.30 ±34.60	213.14 ± 4.69
p_b^1	b	1005.60	6	18	77	111.29 ±40.57	89.60 ±32.40	186.46 ±3.95	1006.54	3	6	30	670.73 ±219.93	94.90 ±31.30	1062.70 ±10.16	1006.69	4	4	24	876.11 ±325.26	91.50 ±34.20	1438.81 ±12.53
	w	1012.55	8	20	14	135.64 ±75.51	74.10 ±41.20	274.74 ± 9.53	1102.43	388	20	97	212.50 ±25.39	143.00 ±15.50	224.09 ± 7.54	1062.14	201	20	99	142.22 ±12.78	145.00 ±12.00	148.15 ± 3.26
p_b^2	b	***1004.86**	5	19	77	99.07 ±32.84	86.70 ±28.70	171.67 ±4.17	**1006.40**	3	5	37	755.69 ±270.61	89.00 ±32.00	1277.33 ±12.19	**1006.35**	5	3	67	847.08 ±256.34	99.10 ±30.00	1288.50 ±9.66
	w	1012.23	6	20	6	129.65 ±79.18	65.00 ±39.70	302.71 ± 11.77	1125.09	294	20	93	225.11 ±21.32	146.00 ±13.20	233.24 ± 7.53	1108.92	256	20	98	145.06 ±5.22	150.00 ±3.32	145.77 ± 3.78
p_b^3	b	1007.24	4	2	1	2001.57 ±1011.74	76.90 ±38.90	3912.91 ±39.79	1007.11	4	6	7	712.92 ±299.46	88.70 ±37.40	1209.02 ±11.62	1007.11	4	6	2	676.31 ±304.82	80.00 ±36.20	1272.72 ±10.62
	w	1008.96	4	20	100	108.97 ±35.31	94.00 ±30.60	174.95 ± 4.71	1008.54	4	19	95	128.77 ±31.61	99.80 ±24.60	193.38 ± 4.84	1008.49	3	20	88	114.47 ±35.28	95.00 ±29.60	181.71 ± 4.66
p_b^4	b	1007.24	4	1	1	3920.41 ±2019.98	75.70 ±39.00	7787.11 ±103.25	1006.74	3	6	17	673.49 ±241.35	91.60 ±32.70	1107.44 ±11.16	1006.82	3	4	10	980.12 ±421.02	86.30 ±37.20	1709.86 ±17.64
	w	1011.68	8	18	64	120.59 ±57.17	81.50 ±38.60	222.18 ± 5.29	1012.44	114	19	98	176.75 ±32.57	127.00 ±23.00	210.04 ± 5.48	1008.37	4	20	99	128.58 ±25.41	116.00 ±22.70	167.62 ± 3.82
p_b^5	b	1005.82	6	18	73	114.83 ±36.09	95.50 ±30.00	181.20 ±3.82	1006.56	3	5	21	799.83 ±284.53	91.40 ±32.50	1315.89 ±11.75	1006.61	4	5	29	700.83 ±245.66	99.40 ±34.90	1059.86 ±8.50
	w	1012.82	6	18	13	148.04 ±84.29	71.90 ±40.60	309.94 ± 10.95	1085.88	332	20	96	208.82 ±20.76	144.00 ±12.60	219.04 ± 6.94	1066.84	202	20	98	137.03 ±8.42	148.00 ±7.62	140.01 ± 3.34
p_b^6	b	1007.24	4	1	1	3920.41 ±2019.98	75.70 ±38.90	7759.55 ±103.02	1006.70	3	5	30	774.09 ±276.39	96.20 ±34.40	1210.08 ±12.11	1006.78	3	6	8	635.77 ±268.48	84.20 ±35.50	1136.54 ±10.89
	w	1011.71	7	19	33	135.08 ±62.39	84.80 ±39.40	240.66 ± 6.76	1020.63	92	20	98	171.51 ±28.24	132.00 ±21.60	194.96 ± 5.32	1017.02	147	19	98	142.73 ±20.73	134.00 ±19.60	160.84 ± 3.32
p_b^7	b	1007.23	4	1	1	3986.68 ±2038.43	77.20 ±39.60	7758.00 ±85.80	1006.89	2	4	3	1058.76 ±469.19	85.00 ±37.80	1873.53 ±14.85	1006.91	3	6	6	710.99 ±287.37	90.20 ±36.40	1185.49 ±13.59
	w	1008.72	3	20	99	91.14 ±34.82	87.90 ±33.50	156.40 ± 3.45	1008.34	4	19	98	95.93 ±39.51	86.80 ±35.60	166.55 ± 3.75	1008.37	4	15	100	116.94 ±45.16	84.40 ±32.50	209.00 ± 3.30
p_b^8	b	1004.90	5	20	67	100.69 ±34.25	92.90 ±32.00	163.45 ±3.57	1006.42	4	6	25	687.27 ±225.05	94.30 ±31.10	1095.98 ±9.30	1006.51	3	3	65	791.35 ±268.73	94.00 ±32.00	1266.97 ±11.99
	w	1012.00	6	20	9	153.06 ±76.73	79.80 ±40.00	287.92 ± 10.89	1122.50	280	20	92	226.78 ±15.53	148.00 ±8.53	231.11 ± 7.31	1114.40	279	20	96	132.91 ±4.98	150.00 ±3.96	133.87 ± 3.20
p_b^9	b	1005.98	7	20	75	93.41 ±25.04	100.00 ±27.00	140.96 ±2.82	1006.61	3	5	30	779.15 ±247.94	96.30 ±30.40	1217.33 ±13.06	1006.81	4	6	4	681.19 ±282.04	85.50 ±35.60	1200.18 ±12.34
	w	1012.79	9	20	20	137.58 ±60.65	87.00 ±38.00	238.52 ± 8.13	1092.27	306	20	78	208.55 ±20.84	144.00 ±13.30	219.05 ± 7.51	1088.31	232	20	95	123.41 ±8.07	149.00 ±9.15	125.25 ± 2.77

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.20: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra200_12* [Dav06b]. Stopping criterion, $T = 0.2[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	1203.89	2	1	1	3407.01 ±1948.06	88.40 ±50.50	7727.27 ±82.42	1203.41	3	7	9	480.49 ±265.99	94.00 ±52.10	1024.70 ±8.87	1203.56	3	4	6	922.82 ±487.77	101.00 ±53.40	1837.13 ±12.99
	w	1205.24	4	18	87	107.13 ±63.26	86.60 ±51.30	248.06 ± 4.52	1204.13	3	20	98	96.63 ±49.03	89.90 ±45.30	215.37 ± 4.78	1204.23	3	20	81	98.64 ±49.36	88.70 ±44.50	223.67 ± 4.95
p_b^1	b	1202.49	3	20	91	85.62 ±38.17	92.10 ±41.20	185.91 ±3.66	1203.04	2	8	59	398.88 ±170.98	103.00 ±44.10	775.20 ±8.06	1203.61	3	11	5	295.23 ±151.07	91.50 ±46.80	647.13 ±8.27
	w	1207.96	9	19	15	132.20 ±84.89	85.20 ±54.90	313.50 ± 8.98	1204.23	44	20	98	170.19 ±49.35	132.00 ±38.20	259.26 ± 6.31	1205.79	8	20	91	152.92 ±26.04	164.00 ±28.10	187.51 ± 3.56
p_b^2	b	***1202.27**	3	20	91	78.35 ±41.07	86.60 ±45.90	182.04 ±3.40	**1202.93**	2	8	34	408.10 ±211.58	97.30 ±50.40	841.60 ±7.81	1203.57	2	8	6	416.61 ±226.99	92.70 ±50.40	900.81 ±7.80
	w	1207.98	6	20	7	122.53 ±96.30	77.50 ±60.90	318.20 ± 9.55	1205.60	31	20	90	203.98 ±45.81	150.00 ±33.90	273.25 ± 6.48	1206.71	50	20	96	156.38 ±21.36	173.00 ±23.90	181.11 ± 3.30
p_b^3	b	1203.71	3	11	13	329.85 ±162.96	106.00 ±52.00	625.65 ±9.68	1203.74	4	13	7	243.33 ±140.20	90.50 ±52.20	539.50 ±8.59	1203.74	3	8	4	375.31 ±237.64	80.30 ±51.00	937.49 ±6.94
	w	1204.88	3	20	99	115.47 ±48.60	110.00 ±46.20	210.85 ± 4.41	1204.34	2	12	93	200.55 ±85.89	104.00 ±45.30	384.53 ± 5.12	1204.47	2	19	84	110.13 ±49.96	95.50 ±43.00	232.52 ± 4.66
p_b^4	b	1203.90	2	2	1	1697.22 ±973.83	87.50 ±50.00	3884.23 ±32.41	1203.27	2	6	57	505.11 ±250.22	100.00 ±49.80	1007.53 ±7.30	1203.60	3	12	7	297.49 ±157.65	103.00 ±54.50	579.69 ±10.43
	w	1207.15	6	20	44	111.54 ±62.30	87.50 ±48.70	256.08 ± 5.17	1204.02	2	1	84	2871.53 ±1516.61	92.70 ±49.10	6207.06 ± 64.96	1204.35	2	17	79	154.88 ±58.67	119.00 ±45.60	261.66 ± 4.49
p_b^5	b	1202.51	4	18	97	93.10 ±50.43	95.60 ±51.80	195.12 ±3.05	1203.03	2	7	50	461.47 ±214.01	103.00 ±47.80	899.72 ±6.93	1203.64	3	7	5	468.78 ±271.20	90.10 ±52.10	1042.69 ±8.10
	w	1208.22	5	20	13	114.27 ±83.17	75.80 ±55.20	301.77 ± 6.94	1204.16	2	19	89	170.59 ±52.68	125.00 ±38.60	273.38 ± 6.72	1206.18	23	20	95	143.10 ±23.47	165.00 ±26.70	174.08 ± 2.99
p_b^6	b	1203.89	2	1	1	3407.01 ±1948.06	88.20 ±50.40	7734.33 ±75.77	1203.18	2	5	47	552.41 ±296.54	90.10 ±48.40	1230.89 ±8.13	1203.58	3	8	9	443.45 ±223.15	103.00 ±52.20	861.06 ±7.03
	w	1207.32	7	20	27	113.18 ±72.88	84.10 ±54.50	270.41 ± 6.74	1204.02	2	1	82	2871.53 ±1516.61	92.50 ±49.00	6221.81 ± 57.56	1204.67	3	20	94	122.50 ±35.63	131.00 ±38.20	187.34 ± 3.41
p_b^7	b	1203.90	2	1	1	3344.81 ±1907.99	86.70 ±49.60	7727.00 ±70.85	1203.46	2	4	7	744.13 ±480.98	82.50 ±53.50	1806.65 ±13.00	1203.64	3	13	4	260.29 ±137.97	96.30 ±51.40	543.83 ±9.62
	w	1204.84	3	20	3	156.90 ±90.82	95.00 ±55.20	331.96 ± 8.34	1204.40	2	15	99	115.36 ±62.11	88.70 ±47.90	261.01 ± 3.52	1204.37	4	18	87	103.48 ±55.44	93.70 ±50.50	221.43 ± 4.32
p_b^8	b	1202.41	4	17	90	98.36 ±57.77	91.70 ±54.00	215.36 ±3.39	1202.95	2	12	49	278.45 ±109.43	107.00 ±42.30	522.78 ±7.79	**1203.54**	3	6	5	533.92 ±314.83	88.30 ±52.10	1212.98 ±9.99
	w	1207.20	6	20	7	129.05 ±92.40	81.60 ±58.40	317.73 ± 9.74	1204.88	35	20	79	203.67 ±50.04	149.00 ±36.30	273.60 ± 7.14	1209.22	63	20	98	140.80 ±18.45	175.00 ±23.00	161.96 ± 2.72
p_b^9	b	1202.61	4	18	95	85.69 ±43.40	98.40 ±50.00	174.78 ±2.94	1203.05	2	8	42	419.02 ±179.29	109.00 ±46.50	770.89 ±6.60	1203.64	3	5	3	734.85 ±397.80	98.90 ±53.70	1488.59 ±11.80
	w	1208.33	6	19	14	118.74 ±76.26	78.40 ±50.50	303.73 ± 6.89	1204.35	42	19	100	167.21 ±47.27	131.00 ±37.70	254.85 ± 5.94	1207.76	55	20	98	133.86 ±18.78	178.00 ±24.90	151.31 ± 2.68

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.21: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra200_16* [Dav06b]. Stopping criterion, $T = 0.2[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\tilde{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\tilde{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\tilde{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	1206.76	3	1	1	3576.58 ±1823.31	101.00 ±51.30	7121.33 ±67.92	1206.31	3	3	28	1123.29 ±502.07	111.00 ±49.40	2034.37 ±15.37	1206.42	3	3	4	1236.09 ±574.98	108.00 ±50.30	2293.92 ±17.32
	w	1208.86	4	20	88	107.70 ±46.17	111.00 ±48.00	193.71 ± 5.31	1207.30	3	20	98	114.71 ±40.65	120.00 ±42.70	191.29 ± 5.37	1207.35	3	19	85	117.35 ±46.49	114.00 ±45.30	205.82 ± 5.84
p_b^1	b	1206.76	3	1	1	3576.58 ±1823.31	101.00 ±51.40	7108.59 ±67.11	1205.91	3	6	34	655.58 ±181.68	131.00 ±36.30	1006.07 ±9.48	1206.35	4	7	11	494.86 ±197.23	111.00 ±44.60	891.92 ±9.28
	w	1215.12	166	20	81	141.32 ±24.72	164.00 ±28.10	173.44 ± 4.71	1221.97	282	18	88	218.08 ±34.17	173.00 ±26.70	253.56 ± 6.58	1221.87	200	19	100	145.93 ±25.70	174.00 ±30.70	168.31 ± 3.47
p_b^2	b	1206.21	10	16	98	150.20 ±38.42	147.00 ±37.20	205.94 ±4.78	**1205.72**	3	7	38	569.99 ±170.95	132.00 ±39.40	865.47 ±9.67	**1206.27**	3	4	5	945.24 ±396.46	112.00 ±47.10	1690.97 ±12.03
	w	1218.36	372	19	54	165.52 ±32.14	160.00 ±31.50	207.79 ± 5.24	1284.06	348	19	100	230.13 ±38.23	177.00 ±28.20	261.05 ± 12.27	1224.83	220	19	100	148.15 ±21.62	178.00 ±25.30	166.82 ± 5.21
p_b^3	b	1206.73	3	2	15	1845.05 ±892.14	114.00 ±55.20	3255.70 ±27.04	1206.70	3	2	2	1837.22 ±863.87	104.00 ±49.10	3541.54 ±30.95	1206.70	3	3	6	1255.25 ±581.24	110.00 ±51.10	2279.88 ±19.76
	w	1208.18	4	20	100	125.67 ±38.73	136.00 ±41.30	185.33 ± 5.30	1207.72	4	19	95	132.96 ±40.35	130.00 ±39.80	204.77 ± 5.60	1207.64	4	18	94	124.85 ±44.63	121.00 ±43.30	208.20 ± 5.29
p_b^4	b	1206.76	3	1	1	3576.58 ±1823.31	101.00 ±51.30	7118.83 ±63.87	1206.08	4	7	22	577.39 ±187.35	134.00 ±43.60	865.77 ±10.57	1206.35	4	4	8	987.63 ±398.37	120.00 ±48.40	1647.17 ±13.41
	w	1213.12	8	20	87	121.89 ±39.93	132.00 ±42.70	185.23 ± 5.00	1208.66	116	20	94	171.08 ±25.55	165.00 ±24.30	207.63 ± 6.25	1207.40	5	18	80	142.32 ±38.77	133.00 ±36.30	214.44 ± 5.62
p_b^5	b	1206.76	3	1	1	3576.58 ±1823.31	100.00 ±51.30	7128.78 ±68.99	1205.93	2	6	38	677.87 ±183.28	138.00 ±37.60	986.34 ±11.25	1206.41	4	3	10	1174.11 ±494.04	110.00 ±46.50	2133.63 ±16.94
	w	1219.38	323	19	95	140.39 ±20.68	171.00 ±24.30	164.95 ± 3.26	1223.34	320	20	87	195.64 ±31.16	172.00 ±26.20	228.35 ± 6.35	1219.31	161	19	93	138.86 ±22.01	168.00 ±25.70	166.48 ± 4.23
p_b^6	b	1206.76	3	1	1	3576.58 ±1823.31	101.00 ±51.30	7126.70 ±62.55	1206.10	2	5	20	760.94 ±275.02	124.00 ±45.00	1227.21 ±9.48	1206.30	4	8	7	453.37 ±197.07	113.00 ±49.40	801.61 ±9.65
	w	1214.33	135	20	98	123.73 ±28.69	153.00 ±34.80	163.46 ± 4.19	1208.73	41	19	92	180.47 ±31.20	164.00 ±28.00	220.92 ± 6.23	1209.46	137	20	98	127.52 ±26.22	156.00 ±32.20	164.50 ± 3.99
p_b^7	b	1206.76	3	1	1	3576.58 ±1823.31	101.00 ±51.40	7124.81 ±81.77	1206.39	4	4	7	865.19 ±438.80	104.00 ±52.80	1664.59 ±13.91	1206.38	3	4	7	907.22 ±408.67	110.00 ±49.40	1649.93 ±14.20
	w	1208.24	4	14	4	229.09 ±103.74	107.00 ±49.00	427.71 ± 10.93	1207.58	3	20	82	108.71 ±41.60	118.00 ±45.20	184.65 ± 4.86	1207.64	4	19	99	102.26 ±39.88	117.00 ±45.60	175.26 ± 4.16
p_b^8	b	***1205.02**	9	19	100	118.95 ±29.68	147.00 ±36.70	162.35 ±3.82	1205.74	3	6	29	662.37 ±205.87	129.00 ±39.90	1029.02 ±9.54	1206.29	4	4	5	981.50 ±384.92	117.00 ±46.10	1678.95 ±14.09
	w	1213.26	291	19	78	129.93 ±34.86	146.00 ±38.50	178.61 ± 4.89	1264.66	279	19	74	226.75 ±36.09	180.00 ±27.70	253.17 ± 7.94	1232.04	192	20	100	127.91 ±20.40	179.00 ±28.00	143.85 ± 3.12
p_b^9	b	1206.76	3	1	1	3576.58 ±1823.31	101.00 ±51.30	7116.39 ±68.00	1205.98	2	5	49	670.05 ±237.46	120.00 ±42.50	1121.48 ±10.73	1206.45	3	4	9	859.70 ±346.82	110.00 ±44.50	1569.14 ±13.09
	w	1220.01	351	20	87	123.39 ±19.39	168.00 ±25.60	147.81 ± 3.63	1228.24	350	19	83	207.61 ±29.54	176.00 ±24.40	236.19 ± 8.25	1221.02	200	20	91	124.34 ±18.22	174.00 ±25.50	143.88 ± 3.57

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.22: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra250_12* [Dav06b]. Stopping criterion, $T = 0.25[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	1403.39	2	1	1	2816.60 ±1747.68	100.00 ±62.40	7034.01 ±58.37	1402.98	2	5	7	590.00 ±338.44	110.00 ±63.10	1345.42 ±8.41	1403.18	2	4	9	747.94 ±483.55	113.00 ±73.00	1656.42 ±11.11
	w	1404.55	3	17	94	112.50 ±70.44	109.00 ±68.00	259.03 ± 5.84	1403.53	1	18	99	100.73 ±61.67	104.00 ±63.80	243.79 ± 5.02	1403.69	2	7	95	220.31 ±148.45	81.20 ±55.10	678.84 ± 5.05
p_b^1	b	1403.39	2	1	1	2816.60 ±1747.68	101.00 ±62.50	7029.99 ±58.06	1402.65	2	8	52	314.37 ±174.13	106.00 ±58.30	745.90 ±6.99	1403.18	2	6	4	518.13 ±305.13	115.00 ±67.70	1128.13 ±7.43
	w	1407.24	7	19	16	93.02 ±78.36	80.70 ±68.10	291.46 ± 7.17	1403.49	1	19	98	156.94 ±47.46	149.00 ±45.20	264.94 ± 6.01	1404.76	4	20	92	134.01 ±31.05	180.00 ±41.50	186.74 ± 3.34
p_b^2	b	1403.39	2	3	1	968.88 ±597.05	103.00 ±63.70	2346.36 ±13.58	***1402.59**	1	11	50	223.72 ±119.45	104.00 ±55.50	536.33 ±8.53	1403.20	2	11	4	237.77 ±145.03	101.00 ±61.40	591.52 ±9.18
	w	1407.22	6	20	9	95.45 ±94.23	82.40 ±81.40	289.25 ± 7.31	1404.72	2	20	99	172.46 ±42.17	169.00 ±41.50	257.18 ± 6.58	1404.50	3	20	86	133.87 ±29.38	179.00 ±38.80	188.02 ± 3.90
p_b^3	b	**1403.28**	2	3	11	879.44 ±583.07	99.30 ±65.90	2219.80 ±14.98	1403.25	2	5	10	569.93 ±347.79	107.00 ±65.40	1331.76 ±8.67	1403.27	2	3	2	929.34 ±590.88	99.80 ±63.60	2329.06 ±12.87
	w	1404.08	2	20	95	121.03 ±51.91	140.00 ±60.20	217.22 ± 5.74	1403.78	3	16	98	111.87 ±49.46	101.00 ±45.20	276.52 ± 5.27	1403.81	2	18	89	106.41 ±57.56	110.00 ±59.80	242.65 ± 5.18
p_b^4	b	1403.39	2	1	1	2816.60 ±1747.68	100.00 ±62.40	7024.17 ±61.43	1402.81	2	6	48	440.22 ±224.58	111.00 ±57.10	989.00 ±7.16	1403.20	2	5	4	642.66 ±406.32	118.00 ±75.10	1358.92 ±8.88
	w	1406.94	5	20	71	91.23 ±63.85	102.00 ±72.00	225.09 ± 5.00	1403.48	2	20	1	128.02 ±70.61	102.00 ±57.00	314.01 ± 9.32	1403.87	2	18	99	112.79 ±51.12	123.00 ±55.30	230.01 ± 4.13
p_b^5	b	1403.39	2	1	1	2816.60 ±1747.68	101.00 ±62.50	7022.39 ±55.41	1402.61	1	9	50	297.48 ±156.63	115.00 ±59.90	650.64 ±8.14	1403.23	2	4	7	678.46 ±423.22	103.00 ±64.50	1650.38 ±11.04
	w	1407.33	6	20	16	94.28 ±73.06	86.20 ±66.90	273.83 ± 6.97	1403.50	2	17	2	136.37 ±86.76	92.40 ±58.70	369.20 ± 9.68	1404.63	3	20	89	128.84 ±31.34	175.00 ±43.20	184.36 ± 2.91
p_b^6	b	1403.40	2	1	1	2753.86 ±1707.74	97.80 ±60.80	7038.21 ±59.17	1402.79	2	9	81	277.41 ±136.76	118.00 ±58.40	589.63 ±6.14	**1403.15**	2	4	6	681.41 ±439.34	103.00 ±66.40	1658.76 ±12.22
	w	1406.95	5	20	45	96.67 ±74.08	103.00 ±79.10	236.15 ± 5.01	1403.48	2	17	1	148.67 ±83.55	101.00 ±55.80	373.02 ± 9.15	1404.23	2	17	96	121.08 ±42.42	134.00 ±46.60	226.11 ± 3.92
p_b^7	b	1403.39	2	1	1	2816.60 ±1747.68	100.00 ±62.20	7030.37 ±62.06	1403.03	2	6	7	484.29 ±287.80	110.00 ±65.10	1106.12 ±8.53	**1403.15**	2	8	10	336.81 ±217.95	105.00 ±68.40	801.72 ±8.88
	w	1404.23	2	17	4	149.68 ±102.08	104.00 ±70.30	361.27 ± 9.57	1403.73	2	20	77	81.61 ±50.56	94.40 ±58.70	216.81 ± 5.36	1403.80	2	20	81	82.09 ±45.20	99.00 ±54.70	207.58 ± 4.48
p_b^8	b	1403.40	2	1	1	2753.86 ±1707.74	98.30 ±61.00	7014.74 ±59.22	***1402.59**	2	10	54	268.76 ±141.10	115.00 ±60.70	586.42 ±7.94	1403.20	2	8	3	339.51 ±218.27	101.00 ±64.90	842.55 ±8.13
	w	1406.84	5	20	9	87.02 ±77.92	75.30 ±67.10	288.25 ± 7.12	1403.93	2	20	90	175.27 ±47.55	168.00 ±45.50	260.20 ± 6.10	1404.57	3	20	92	125.32 ±26.12	187.00 ±39.20	167.93 ± 3.10
p_b^9	b	1403.39	2	1	2	2816.60 ±1747.68	101.00 ±62.60	6996.62 ±54.96	1402.70	2	6	40	470.30 ±255.51	118.00 ±64.10	997.88 ±7.48	1403.23	2	8	4	374.48 ±224.69	113.00 ±67.70	831.01 ±7.93
	w	1407.31	4	19	15	88.84 ±81.39	79.00 ±72.40	283.70 ± 7.74	1403.51	3	20	99	144.07 ±46.20	151.00 ±48.50	238.60 ± 5.27	1404.91	3	20	90	120.11 ±25.21	185.00 ±39.00	163.33 ± 3.38

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.23: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra250_16* [Dav06b]. Stopping criterion, $T = 0.25[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	1405.23	3	2	1	1610.58 ±959.84	123.00 ±73.30	3273.22 ±22.74	1404.82	2	5	8	649.28 ±316.53	132.00 ±64.30	1231.97 ±11.08	1404.90	3	4	9	735.82 ±379.88	120.00 ±62.40	1529.59 ±12.39
	w	1406.61	3	18	99	118.08 ±52.53	140.00 ±62.50	211.82 ± 4.83	1405.55	3	18	90	111.90 ±48.40	126.00 ±54.40	222.04 ± 5.98	1405.60	2	20	92	96.88 ±44.37	124.00 ±57.40	196.24 ± 5.63
p_b^1	b	1404.04	4	19	99	88.87 ±35.69	130.00 ±52.60	171.33 ±3.56	1404.49	3	7	50	424.78 ±157.38	137.00 ±51.00	778.50 ±9.06	1404.77	3	4	77	573.25 ±242.08	141.00 ±59.90	1016.81 ±7.42
	w	1409.68	8	20	18	137.83 ±52.31	142.00 ±54.10	244.32 ± 7.64	1408.81	149	17	91	186.29 ±42.57	177.00 ±40.00	263.29 ± 8.93	1407.78	177	20	89	128.29 ±23.66	188.00 ±35.70	170.94 ± 4.16
p_b^2	b	***1403.61**	4	19	98	90.50 ±37.63	136.00 ±57.40	167.00 ±3.07	1404.43	3	8	31	391.00 ±165.92	139.00 ±58.70	706.57 ±9.50	**1404.54**	3	4	75	572.60 ±213.04	145.00 ±53.90	992.86 ±6.56
	w	1409.76	7	20	9	135.90 ±63.29	133.00 ±62.20	257.10 ± 7.94	1422.47	189	19	88	189.85 ±42.08	197.00 ±43.90	241.71 ± 6.96	1408.64	183	20	89	129.19 ±25.81	191.00 ±38.80	169.50 ± 3.71
p_b^3	b	1405.10	3	2	14	1456.23 ±751.14	119.00 ±61.50	3062.49 ±19.77	1405.08	2	4	9	749.64 ±362.80	121.00 ±58.80	1548.00 ±12.83	1405.12	2	3	4	1038.57 ±604.72	122.00 ±71.00	2132.29 ±16.08
	w	1406.23	4	20	92	116.30 ±41.75	149.00 ±52.70	196.83 ± 5.79	1405.77	2	16	88	138.25 ±51.94	137.00 ±51.80	252.88 ± 5.59	1405.76	2	20	97	100.01 ±38.75	134.00 ±51.10	188.30 ± 5.38
p_b^4	b	1405.23	3	1	1	3196.51 ±1940.32	123.00 ±74.20	6536.25 ±48.62	1404.64	3	6	64	477.03 ±190.72	139.00 ±55.80	861.02 ±9.11	1404.78	2	5	30	579.75 ±241.93	135.00 ±56.00	1077.28 ±10.64
	w	1408.71	5	20	46	127.04 ±47.92	143.00 ±53.60	222.26 ± 7.50	1405.49	3	20	1	134.31 ±69.38	121.00 ±62.00	279.08 ± 9.36	1405.52	3	16	100	137.07 ±45.59	149.00 ±49.90	231.29 ± 6.31
p_b^5	b	1404.05	4	20	98	95.77 ±34.78	150.00 ±55.40	159.89 ±2.70	1404.50	2	9	61	337.87 ±107.37	149.00 ±46.90	565.76 ±11.04	1404.77	2	4	73	569.31 ±223.79	142.00 ±55.90	1001.82 ±9.58
	w	1409.90	7	20	17	138.49 ±63.96	140.00 ±64.90	248.48 ± 7.95	1408.30	151	19	87	169.46 ±40.24	182.00 ±43.30	233.57 ± 7.77	1407.83	176	20	99	117.53 ±26.25	184.00 ±40.60	160.60 ± 3.77
p_b^6	b	1405.23	3	2	1	1610.58 ±959.84	123.00 ±73.50	3278.74 ±22.00	1404.62	3	6	32	473.24 ±201.49	129.00 ±55.10	921.76 ±9.36	1404.82	3	3	6	955.46 ±542.74	116.00 ±66.10	2057.75 ±14.39
	w	1408.90	5	20	24	138.97 ±55.95	148.00 ±60.30	235.24 ± 6.87	1405.50	3	20	1	133.01 ±67.75	119.00 ±60.10	280.44 ± 9.06	1405.90	21	19	98	124.46 ±34.51	171.00 ±46.70	182.55 ± 4.53
p_b^7	b	1405.23	3	3	1	1091.42 ±628.35	125.00 ±72.30	2178.96 ±13.21	1404.85	2	3	7	936.14 ±508.69	113.00 ±61.50	2074.88 ±14.88	1404.89	2	6	8	488.83 ±251.47	122.00 ±63.20	1002.93 ±11.42
	w	1406.27	5	20	4	147.24 ±67.43	131.00 ±59.70	281.31 ± 9.15	1405.75	3	19	95	96.69 ±45.24	128.00 ±59.80	189.92 ± 4.79	1405.74	2	15	97	120.88 ±50.71	128.00 ±54.00	237.35 ± 6.21
p_b^8	b	1403.69	5	20	90	77.62 ±36.00	119.00 ±54.80	163.77 ±4.18	**1404.42**	3	7	25	469.82 ±173.60	142.00 ±52.60	827.40 ±9.98	1404.68	3	4	77	558.80 ±217.04	148.00 ±57.40	947.63 ±7.04
	w	1409.22	6	18	9	162.74 ±70.22	142.00 ±61.90	286.16 ± 9.13	1414.70	228	18	96	196.41 ±39.58	196.00 ±39.70	251.24 ± 6.78	1407.65	145	20	77	129.07 ±26.90	192.00 ±40.30	168.41 ± 4.30
p_b^9	b	1404.24	4	18	85	97.28 ±36.50	142.00 ±53.10	171.91 ±3.73	1404.53	3	5	46	562.49 ±241.14	130.00 ±56.00	1086.22 ±8.79	1404.87	2	4	78	517.64 ±202.63	146.00 ±57.40	887.37 ±6.43
	w	1409.96	7	19	17	135.27 ±59.48	134.00 ±59.00	252.25 ± 6.87	1407.86	172	20	93	157.16 ±33.53	185.00 ±38.20	211.88 ± 6.31	1407.41	127	19	91	120.30 ±21.61	192.00 ±34.30	157.27 ± 3.16

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.24: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra300_12* [Dav06b]. Stopping criterion, $T = 0.3[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	1603.05	2	2	1	1308.66 ±925.11	122.00 ±85.90	3230.74 ±18.45	1602.81	2	6	14	404.69 ±244.31	121.00 ±72.90	1005.45 ±7.26	1602.96	2	3	37	817.29 ±455.26	131.00 ±72.90	1879.06 ±11.80
	w	1604.22	2	16	91	106.85 ±66.00	116.00 ±71.30	277.47 ± 4.78	1603.23	2	18	82	100.46 ±57.95	119.00 ±68.80	252.83 ± 4.80	1603.36	2	15	88	122.21 ±75.29	124.00 ±76.60	297.12 ± 4.94
p_b^1	b	1603.04	4	17	94	92.07 ±50.76	123.00 ±67.90	226.20 ±4.04	1602.46	1	11	60	198.74 ±116.12	122.00 ±71.20	488.30 ±7.96	1602.95	2	9	5	316.35 ±165.74	142.00 ±74.60	669.07 ±9.02
	w	1606.87	5	19	14	102.22 ±78.14	110.00 ±83.80	279.53 ± 6.51	1603.19	2	17	2	138.55 ±88.95	121.00 ±78.30	343.37 ± 6.91	1604.00	3	19	78	127.25 ±42.01	184.00 ±60.20	208.18 ± 3.98
p_b^2	b	1602.93	4	14	92	111.95 ±72.24	123.00 ±79.70	274.22 ±4.05	1602.41	2	13	52	194.77 ±104.16	140.00 ±74.30	416.93 ±7.71	1602.95	2	8	5	335.49 ±183.26	132.00 ±72.10	761.88 ±6.90
	w	1606.79	5	20	10	86.65 ±71.66	96.40 ±79.60	269.95 ± 6.65	1603.16	2	18	1	133.73 ±88.56	123.00 ±81.90	325.34 ± 8.15	1603.99	3	20	82	118.77 ±36.68	190.00 ±58.90	188.70 ± 3.89
p_b^3	b	1603.01	2	5	14	559.93 ±318.99	139.00 ±79.10	1212.48 ±8.53	1603.03	2	6	10	452.27 ±303.24	133.00 ±89.10	1023.81 ±6.98	1603.02	2	6	10	469.09 ±273.71	138.00 ±80.40	1021.81 ±8.21
	w	1603.77	2	18	98	122.51 ±53.69	155.00 ±67.50	239.33 ± 5.05	1603.42	2	20	80	91.15 ±48.96	119.00 ±64.90	228.78 ± 4.98	1603.49	1	18	79	98.78 ±60.74	120.00 ±74.50	249.06 ± 5.24
p_b^4	b	1603.06	2	1	1	2507.71 ±1802.73	117.00 ±83.90	6455.78 ±46.23	1602.59	1	8	32	326.97 ±194.78	139.00 ±82.20	709.12 ±6.48	**1602.91**	2	12	7	210.03 ±124.43	131.00 ±77.30	482.71 ±8.53
	w	1606.24	4	20	86	103.12 ±60.84	144.00 ±84.90	214.97 ± 4.35	1603.19	2	17	2	138.55 ±88.95	122.00 ±78.60	344.83 ± 7.70	1603.46	2	18	79	112.48 ±54.97	139.00 ±68.00	242.67 ± 5.06
p_b^5	b	1603.06	2	1	1	2507.71 ±1802.73	117.00 ±83.90	6452.05 ±44.47	1602.46	1	7	64	329.37 ±201.25	125.00 ±76.60	789.48 ±6.51	1602.96	2	4	5	652.38 ±367.87	126.00 ±71.40	1552.50 ±10.40
	w	1606.86	5	20	14	93.22 ±70.84	105.00 ±79.90	265.67 ± 6.45	1603.18	2	13	2	162.73 ±105.06	107.00 ±69.40	457.06 ± 10.09	1604.07	2	19	83	112.17 ±39.02	172.00 ±59.50	196.94 ± 3.27
p_b^6	b	1603.05	2	3	1	874.72 ±614.32	122.00 ±85.90	2151.47 ±12.60	1602.53	1	7	75	322.64 ±186.05	129.00 ±74.50	750.89 ±6.54	1602.92	3	9	4	298.86 ±179.90	134.00 ±80.80	671.45 ±9.46
	w	1606.52	5	20	44	101.71 ±62.63	133.00 ±82.10	230.81 ± 4.17	1603.19	2	19	2	129.93 ±82.19	128.00 ±81.80	306.23 ± 7.50	1603.68	2	15	94	116.26 ±52.99	133.00 ±60.50	262.55 ± 4.08
p_b^7	b	1603.06	2	1	1	2507.71 ±1802.73	117.00 ±83.90	6456.85 ±40.12	1602.83	2	4	5	640.29 ±429.19	123.00 ±82.40	1563.36 ±9.97	**1602.91**	2	4	7	645.29 ±437.38	126.00 ±85.50	1535.49 ±10.40
	w	1603.97	2	16	3	143.80 ±100.19	119.00 ±83.00	362.47 ± 7.63	1603.46	2	15	100	102.31 ±71.95	113.00 ±79.80	271.50 ± 4.64	1603.51	2	19	99	73.11 ±51.15	106.00 ±74.40	206.78 ± 3.71
p_b^8	b	**1602.81**	3	16	70	118.23 ±70.55	141.00 ±84.40	251.12 ±3.89	***1602.39**	1	11	45	236.35 ±119.28	142.00 ±71.20	501.64 ±8.73	1602.93	2	8	5	299.55 ±196.30	119.00 ±77.70	758.42 ±7.64
	w	1606.31	5	20	8	100.60 ±65.66	111.00 ±72.20	273.00 ± 6.15	1603.18	2	18	2	125.74 ±84.40	117.00 ±78.60	323.67 ± 7.48	1603.99	2	19	65	128.32 ±40.91	189.00 ±59.70	204.42 ± 3.99
p_b^9	b	1603.05	2	2	1	1308.66 ±925.11	122.00 ±86.00	3229.90 ±18.37	1602.46	2	8	64	277.23 ±169.26	125.00 ±76.00	669.11 ±6.43	1602.94	2	5	7	528.27 ±318.15	132.00 ±79.40	1205.69 ±7.66
	w	1606.87	6	20	16	110.77 ±75.35	131.00 ±88.90	255.69 ± 5.56	1603.21	2	15	3	136.12 ±95.30	105.00 ±73.70	389.06 ± 7.78	1604.13	2	20	80	116.14 ±35.02	199.00 ±60.40	175.70 ± 3.09

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.25: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra300_16* [Dav06b]. Stopping criterion, $T = 0.3[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	1605.26	2	1	1	2397.20 ±1551.00	120.00 ±77.30	6025.47 ±41.43	1604.87	2	5	7	564.95 ±269.96	149.00 ±71.10	1142.28 ±11.46	1605.02	2	8	8	332.84 ±165.93	148.00 ±73.70	677.33 ±10.60
	w	1606.55	4	14	94	127.13 ±68.21	135.00 ±72.50	283.31 ± 6.58	1605.51	3	18	99	107.01 ±50.38	149.00 ±70.10	216.44 ± 5.29	1605.60	3	20	99	96.80 ±41.68	150.00 ±65.40	194.43 ± 4.47
p_b^1	b	1604.53	5	19	96	97.47 ±38.76	167.00 ±66.80	175.05 ±3.20	1604.54	2	8	55	346.46 ±122.68	167.00 ±59.40	624.14 ±10.66	1605.05	3	4	6	723.19 ±334.35	152.00 ±70.70	1426.86 ±12.65
	w	1609.34	6	20	15	109.95 ±65.50	140.00 ±83.40	236.12 ± 6.41	1606.67	107	18	97	171.35 ±39.14	214.00 ±48.10	240.33 ± 6.60	1609.87	116	20	97	136.26 ±25.58	247.00 ±45.80	166.78 ± 3.71
p_b^2	b	***1604.31**	3	18	92	97.93 ±42.59	163.00 ±70.60	181.41 ±3.79	**1604.44**	2	8	44	355.37 ±149.82	167.00 ±69.90	640.02 ±11.71	**1604.84**	3	4	98	516.25 ±196.97	175.00 ±66.80	884.79 ±6.30
	w	1609.35	6	20	9	100.62 ±68.19	123.00 ±83.70	246.53 ± 8.13	1619.88	122	20	67	205.89 ±36.66	269.00 ±47.70	230.52 ± 6.55	1611.71	88	19	85	147.44 ±29.06	250.00 ±49.90	177.17 ± 3.97
p_b^3	b	1605.10	3	4	6	729.37 ±378.48	151.00 ±78.50	1453.20 ±10.60	1605.18	2	3	9	960.28 ±527.66	149.00 ±82.00	1934.53 ±12.00	1605.19	3	4	3	683.33 ±404.45	139.00 ±82.10	1476.79 ±12.64
	w	1606.08	2	19	99	116.70 ±46.39	171.00 ±67.80	206.41 ± 5.73	1605.76	2	20	89	107.96 ±37.57	163.00 ±56.40	198.73 ± 5.61	1605.77	3	18	80	115.59 ±45.72	157.00 ±61.90	220.76 ± 5.19
p_b^4	b	1605.26	2	1	1	2397.20 ±1551.00	119.00 ±77.10	6032.66 ±40.15	1604.66	3	7	67	390.56 ±164.06	170.00 ±71.30	691.37 ±8.09	1605.02	2	3	10	924.84 ±472.50	147.00 ±75.20	1887.87 ±11.91
	w	1608.61	5	20	65	98.03 ±47.75	147.00 ±72.40	201.23 ± 4.47	1605.39	2	13	2	178.40 ±94.65	132.00 ±69.00	409.22 ± 9.53	1605.68	3	18	82	126.93 ±40.22	178.00 ±56.70	214.47 ± 4.99
p_b^5	b	1604.51	3	20	96	84.98 ±33.43	160.00 ±63.80	161.27 ±4.03	1604.57	2	5	41	565.92 ±255.03	161.00 ±72.50	1054.98 ±10.14	1605.01	2	3	12	927.58 ±523.54	151.00 ±85.30	1839.57 ±11.91
	w	1609.39	4	20	14	114.64 ±66.87	145.00 ±84.80	238.07 ± 7.99	1606.23	85	19	98	160.78 ±37.86	210.00 ±49.00	230.49 ± 6.11	1611.88	86	20	95	131.17 ±26.42	245.00 ±48.90	161.18 ± 3.60
p_b^6	b	1605.26	2	1	1	2397.20 ±1551.00	119.00 ±77.30	6040.51 ±39.38	1604.63	3	6	36	478.00 ±200.56	167.00 ±70.00	860.92 ±9.12	1605.00	2	7	6	408.42 ±189.35	156.00 ±72.60	786.66 ±11.52
	w	1608.83	5	20	32	103.25 ±55.89	141.00 ±76.40	221.23 ± 6.39	1605.47	3	19	87	150.47 ±36.94	206.00 ±50.40	219.10 ± 6.05	1606.04	4	20	91	124.90 ±28.93	211.00 ±47.70	178.61 ± 4.22
p_b^7	b	1605.26	2	1	1	2397.20 ±1551.00	120.00 ±77.40	6025.72 ±42.43	1604.93	2	5	10	525.14 ±274.66	141.00 ±73.30	1118.80 ±12.49	1605.04	3	10	11	251.83 ±126.02	147.00 ±73.10	515.14 ±9.95
	w	1606.24	2	17	4	149.69 ±74.91	148.00 ±74.00	302.85 ± 8.14	1605.68	3	20	94	87.77 ±41.98	144.00 ±69.10	184.28 ± 3.99	1605.72	3	19	94	90.02 ±43.30	144.00 ±69.00	188.92 ± 4.00
p_b^8	b	1604.49	4	17	92	101.58 ±46.69	158.00 ±72.80	193.56 ±4.30	1604.45	2	5	46	534.79 ±249.55	152.00 ±71.00	1056.01 ±9.83	1604.98	2	5	5	624.64 ±290.63	165.00 ±77.20	1136.38 ±11.11
	w	1609.17	5	20	9	113.33 ±65.90	139.00 ±80.80	245.49 ± 7.30	1610.10	166	20	99	169.34 ±38.53	228.00 ±51.60	222.48 ± 6.03	1617.23	130	20	93	133.71 ±22.76	264.00 ±45.00	152.98 ± 3.33
p_b^9	b	1604.64	4	19	93	89.63 ±35.96	170.00 ±67.80	158.90 ±3.44	1604.55	3	7	53	389.51 ±163.93	165.00 ±68.60	709.93 ±9.66	1605.07	3	4	6	664.48 ±336.33	141.00 ±71.60	1414.33 ±9.25
	w	1609.43	6	20	17	99.48 ±60.96	128.00 ±77.90	234.48 ± 6.47	1606.48	112	17	98	170.20 ±37.91	205.00 ±46.70	248.78 ± 6.07	1613.84	77	20	96	129.43 ±20.35	260.00 ±40.00	150.17 ± 3.40

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.26: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra350_12* [Dav06b]. Stopping criterion, $T = 0.35[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	1806.20	3	1	1	2915.18 ±1591.77	171.00 ±93.70	5951.20 ±32.67	1805.20	4	4	9	740.60 ±340.99	181.00 ±83.50	1433.17 ±8.67	**1805.53**	3	5	8	589.26 ±281.72	181.00 ±86.60	1138.00 ±8.09
	w	1808.43	3	14	83	171.80 ±77.12	191.00 ±85.40	316.41 ± 4.91	1806.33	3	17	1	171.96 ±79.65	187.00 ±87.20	321.43 ± 6.42	1806.34	3	13	1	220.15 ±101.27	181.00 ±83.20	426.51 ± 7.87
p_b^1	b	1806.19	3	1	3	2972.35 ±1598.14	176.00 ±94.40	5922.24 ±35.49	1804.39	2	7	61	470.12 ±164.17	220.00 ±76.70	751.14 ±5.38	1805.71	4	6	9	540.68 ±217.68	205.00 ±82.00	925.50 ±7.16
	w	1817.35	22	19	48	93.40 ±81.93	139.00 ±122.00	235.27 ± 4.40	1807.41	77	19	97	199.67 ±35.49	285.00 ±50.00	245.72 ± 4.76	1813.46	99	20	98	164.32 ±17.41	321.00 ±33.60	179.90 ± 3.12
p_b^2	b	1806.20	3	1	1	2915.18 ±1591.77	171.00 ±93.80	5952.75 ±30.62	***1804.27**	3	10	55	319.17 ±103.20	216.00 ±69.90	518.85 ±6.61	1805.72	3	6	5	540.59 ±226.50	199.00 ±83.80	948.79 ±6.13
	w	1818.09	23	19	42	96.44 ±84.66	142.00 ±124.00	238.54 ± 4.52	1834.58	130	20	69	213.80 ±47.69	302.00 ±66.80	248.68 ± 5.71	1815.16	77	20	92	163.74 ±15.70	325.00 ±31.20	177.23 ± 3.17
p_b^3	b	**1805.86**	4	6	4	576.06 ±220.95	209.00 ±80.00	968.11 ±6.37	1805.82	4	6	13	482.79 ±235.19	180.00 ±87.90	938.79 ±7.38	1805.87	4	6	3	494.01 ±251.61	178.00 ±90.80	972.31 ±6.79
	w	1807.21	4	20	99	138.89 ±46.22	235.00 ±77.70	208.03 ± 3.49	1806.53	4	16	90	152.82 ±61.22	196.00 ±79.60	271.28 ± 4.17	1806.73	3	20	95	125.60 ±43.14	208.00 ±71.70	212.05 ± 4.01
p_b^4	b	1806.20	3	1	1	2891.77 ±1595.18	170.00 ±93.70	5961.60 ±35.05	1804.70	3	5	69	623.13 ±222.99	211.00 ±75.90	1033.99 ±7.41	1805.65	3	4	16	800.09 ±321.82	203.00 ±82.10	1378.80 ±7.61
	w	1815.55	13	20	98	63.92 ±62.20	111.00 ±108.00	203.24 ± 3.27	1806.35	3	19	1	150.71 ±69.35	185.00 ±85.40	286.66 ± 6.25	1806.74	4	16	98	170.34 ±49.13	233.00 ±67.50	256.32 ± 4.24
p_b^5	b	1806.21	3	1	1	2870.98 ±1567.88	169.00 ±92.30	5956.84 ±28.14	1804.35	3	6	54	561.10 ±201.08	223.00 ±79.90	883.72 ±6.65	1805.74	3	9	7	359.86 ±137.26	208.00 ±79.30	607.68 ±8.96
	w	1817.46	22	19	56	89.56 ±78.89	139.00 ±123.00	225.47 ± 3.78	1807.97	157	20	99	193.62 ±25.72	290.00 ±38.00	234.43 ± 4.80	1813.25	58	19	100	165.40 ±18.84	317.00 ±35.50	183.02 ± 3.00
p_b^6	b	1806.20	3	1	1	2903.15 ±1598.43	171.00 ±93.90	5955.17 ±31.07	1804.59	3	5	96	641.41 ±211.82	227.00 ±75.10	987.87 ±6.62	1805.62	5	10	9	307.48 ±116.86	202.00 ±76.20	535.39 ±7.73
	w	1816.49	17	20	91	55.75 ±59.98	98.70 ±107.00	198.69 ± 3.58	1806.33	3	18	1	163.22 ±75.19	188.00 ±86.10	304.28 ± 6.13	1807.67	5	20	97	153.72 ±25.05	285.00 ±46.10	189.59 ± 3.58
p_b^7	b	1806.19	3	1	1	2935.97 ±1618.10	173.00 ±95.00	5949.67 ±31.58	1805.26	3	4	7	801.47 ±356.40	197.00 ±87.40	1429.95 ±8.73	1805.54	3	4	7	735.49 ±393.71	181.00 ±97.00	1423.71 ±7.73
	w	1807.96	4	20	5	141.53 ±64.36	189.00 ±86.60	263.10 ± 6.00	1806.44	2	19	88	116.02 ±50.44	189.00 ±82.70	215.03 ± 4.01	1806.49	3	19	89	115.33 ±47.53	191.00 ±78.80	211.04 ± 3.67
p_b^8	b	1806.19	3	1	1	2972.35 ±1598.14	175.00 ±94.10	5947.20 ±32.47	1804.30	3	6	53	521.20 ±196.15	205.00 ±77.10	892.80 ±6.78	1805.73	3	2	10	1582.89 ±760.52	198.00 ±94.90	2805.46 ±13.74
	w	1815.58	18	20	27	101.01 ±82.57	152.00 ±124.00	232.90 ± 4.63	1811.48	119	20	97	199.43 ±30.28	295.00 ±44.50	237.56 ± 5.26	1818.94	124	20	97	153.71 ±14.09	327.00 ±29.40	165.05 ± 2.70
p_b^9	b	1806.20	3	1	1	2903.15 ±1598.43	171.00 ±94.00	5954.77 ±36.44	1804.44	3	8	56	405.57 ±127.26	223.00 ±70.30	638.13 ±6.07	1805.80	3	4	5	792.34 ±324.76	194.00 ±79.60	1427.41 ±7.56
	w	1817.49	22	20	82	88.29 ±60.38	168.00 ±115.00	185.21 ± 3.39	1807.18	103	20	90	186.84 ±31.67	282.00 ±47.50	232.05 ± 4.95	1816.25	77	20	95	153.22 ±12.85	330.00 ±27.00	163.46 ± 2.53

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.27: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra350_16* [Dav06b]. Stopping criterion, $T = 0.35[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	1805.09	2	1	1	2412.99 ±1640.42	151.00 ±103.00	5587.59 ±35.80	1804.66	2	4	33	621.18 ±304.27	174.00 ±85.40	1255.73 ±10.79	1804.87	2	5	21	475.48 ±240.61	166.00 ±83.60	1002.89 ±10.59
	w	1806.38	2	17	99	109.49 ±52.05	173.00 ±81.80	222.84 ± 5.02	1805.28	2	18	1	124.60 ±63.40	161.00 ±82.60	272.44 ± 7.05	1805.42	2	17	99	106.69 ±51.49	170.00 ±81.30	221.40 ± 3.94
p_b^1	b	1805.09	2	1	3	2412.99 ±1640.42	153.00 ±104.00	5543.23 ±27.88	1804.34	2	5	44	498.61 ±253.77	176.00 ±89.60	991.79 ±10.30	1804.90	2	2	6	1335.07 ±704.24	173.00 ±91.40	2701.25 ±13.83
	w	1809.43	6	20	19	100.78 ±69.10	160.00 ±111.00	220.10 ± 5.66	1807.26	64	20	89	168.99 ±29.08	285.00 ±48.50	208.29 ± 5.38	1811.53	67	20	99	149.26 ±16.90	322.00 ±36.60	162.95 ± 3.33
p_b^2	b	1805.09	2	1	1	2412.99 ±1640.42	151.00 ±103.00	5583.07 ±30.34	***1804.29**	2	7	51	380.81 ±148.07	195.00 ±76.00	684.69 ±10.56	1804.93	2	2	6	1150.87 ±629.30	150.00 ±81.90	2692.39 ±11.91
	w	1809.36	7	20	12	91.60 ±66.55	139.00 ±100.00	231.70 ± 5.89	1823.62	109	20	92	196.67 ±27.26	319.00 ±43.90	216.79 ± 5.20	1816.72	70	20	98	146.61 ±15.45	328.00 ±35.00	157.43 ± 2.71
p_b^3	b	**1804.91**	3	3	3	876.12 ±494.27	167.00 ±94.40	1834.93 ±12.10	1804.97	2	3	6	819.95 ±471.32	158.00 ±90.90	1817.25 ±10.98	1804.93	2	3	9	814.93 ±439.81	160.00 ±86.40	1789.87 ±11.95
	w	1805.87	3	18	100	118.42 ±45.69	201.00 ±76.80	208.42 ± 3.77	1805.52	2	20	98	113.49 ±34.85	209.00 ±64.40	190.25 ± 3.91	1805.64	1	20	83	105.32 ±35.27	193.00 ±64.70	192.21 ± 4.24
p_b^4	b	1805.09	2	1	1	2412.99 ±1640.42	152.00 ±103.00	5578.23 ±33.58	1804.46	3	4	79	587.42 ±275.86	175.00 ±82.30	1176.47 ±11.01	1804.90	2	2	8	1252.97 ±726.87	163.00 ±94.70	2685.06 ±14.14
	w	1809.04	5	20	81	88.14 ±54.88	164.00 ±102.00	188.38 ± 3.66	1805.27	2	18	1	126.98 ±64.26	165.00 ±83.30	273.86 ± 7.48	1805.78	2	20	90	120.94 ±35.57	230.00 ±67.60	185.14 ± 4.14
p_b^5	b	1805.09	2	1	1	2412.99 ±1640.42	152.00 ±103.00	5577.95 ±29.09	1804.35	3	6	59	446.69 ±197.58	197.00 ±87.60	792.73 ±10.40	1804.97	2	6	3	431.25 ±213.36	172.00 ±85.50	876.27 ±12.18
	w	1809.53	8	20	19	79.92 ±63.48	127.00 ±102.00	219.14 ± 6.06	1806.94	63	20	94	168.23 ±26.70	279.00 ±45.50	211.50 ± 5.84	1812.71	61	20	96	150.78 ±14.32	326.00 ±29.30	162.80 ± 3.71
p_b^6	b	1805.09	2	1	1	2412.99 ±1640.42	152.00 ±103.00	5580.33 ±29.31	1804.42	3	6	53	432.66 ±176.77	193.00 ±79.00	786.76 ±9.76	1804.88	2	2	6	1283.14 ±715.15	167.00 ±93.50	2685.60 ±13.65
	w	1809.12	5	20	41	84.95 ±53.31	143.00 ±89.80	209.24 ± 4.00	1805.27	2	17	1	133.07 ±69.18	160.00 ±83.30	291.24 ± 7.46	1806.30	3	18	100	141.89 ±27.33	263.00 ±49.80	189.56 ± 4.23
p_b^7	b	1805.09	2	1	1	2412.99 ±1640.42	152.00 ±103.00	5580.12 ±28.16	1804.68	2	4	6	575.60 ±314.84	151.00 ±83.10	1333.90 ±12.19	**1804.83**	2	3	9	838.91 ±434.79	167.00 ±86.80	1760.80 ±12.21
	w	1806.03	3	17	4	143.04 ±67.80	178.00 ±83.50	282.70 ± 7.43	1805.43	2	18	80	92.53 ±47.48	154.00 ±78.70	209.53 ± 4.50	1805.49	2	20	97	79.81 ±35.92	157.00 ±71.10	178.80 ± 3.57
p_b^8	b	1805.10	2	1	1	2363.11 ±1620.17	149.00 ±102.00	5572.43 ±30.02	***1804.29**	2	9	53	297.72 ±106.59	201.00 ±71.70	520.03 ±8.24	1804.93	2	3	5	862.62 ±445.38	169.00 ±87.40	1789.09 ±10.41
	w	1808.78	6	20	10	85.21 ±58.38	131.00 ±89.80	227.82 ± 5.55	1813.77	69	20	88	191.57 ±28.96	309.00 ±46.30	217.90 ± 5.07	1820.53	83	20	100	140.24 ±12.01	334.00 ±29.70	147.67 ± 2.64
p_b^9	b	1805.09	2	1	1	2412.99 ±1640.42	151.00 ±103.00	5579.81 ±32.34	1804.38	2	7	62	375.98 ±138.75	200.00 ±74.30	658.99 ±8.77	1804.93	3	4	4	674.68 ±350.78	177.00 ±92.30	1331.28 ±12.46
	w	1809.34	7	20	16	83.65 ±65.28	132.00 ±103.00	223.12 ± 6.30	1807.32	102	20	99	161.82 ±32.23	274.00 ±54.20	207.46 ± 4.25	1818.21	107	20	100	137.95 ±10.74	330.00 ±26.40	147.10 ± 2.86

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.28: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra400_12* [Dav06b]. Stopping criterion, $T = 0.4[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	2003.55	2	1	1	2310.91 ±1378.01	167.00 ±100.00	5518.29 ±26.57	2003.13	2	6	8	438.68 ±243.88	200.00 ±111.00	879.44 ±6.30	2003.27	3	4	9	633.84 ±345.82	192.00 ±105.00	1324.60 ±8.71
	w	2004.93	3	16	93	139.91 ±66.86	217.00 ±104.00	258.81 ± 4.22	2003.65	3	15	2	145.98 ±84.03	172.00 ±99.10	340.73 ± 6.06	2003.80	3	20	92	100.65 ±46.68	196.00 ±91.70	205.34 ± 3.15
p_b^1	b	2003.55	2	1	1	2310.91 ±1378.01	168.00 ±100.00	5517.87 ±26.46	2002.68	2	8	63	296.74 ±125.63	195.00 ±82.60	607.93 ±6.37	2003.30	3	3	5	875.63 ±432.45	196.00 ±96.90	1788.44 ±8.12
	w	2012.84	54	20	55	167.67 ±34.62	328.00 ±68.30	205.29 ± 3.32	2003.63	3	15	2	151.25 ±87.63	178.00 ±103.00	340.45 ± 5.18	2005.36	21	20	100	133.69 ±32.99	300.00 ±74.20	179.18 ± 2.76
p_b^2	b	2003.55	2	1	1	2310.91 ±1378.01	168.00 ±100.00	5513.00 ±25.36	***2002.62**	2	11	51	218.25 ±91.50	198.00 ±83.60	442.12 ±5.20	2003.20	3	7	5	357.31 ±190.19	191.00 ±101.00	748.59 ±6.47
	w	2013.56	82	20	49	171.86 ±38.37	327.00 ±72.80	210.80 ± 4.23	2009.57	58	20	87	177.37 ±51.73	311.00 ±90.80	228.43 ± 3.50	2005.71	47	20	74	133.82 ±35.74	290.00 ±78.10	185.23 ± 2.58
p_b^3	b	**2003.47**	3	2	6	1207.14 ±700.18	178.00 ±103.00	2718.05 ±13.22	2003.45	2	3	10	802.30 ±448.33	180.00 ±101.00	1781.74 ±8.81	2003.46	3	15	16	172.51 ±75.83	212.00 ±92.80	327.85 ±4.98
	w	2004.41	3	19	97	127.51 ±44.82	240.00 ±84.10	214.09 ± 3.63	2003.91	3	14	93	145.64 ±65.10	194.00 ±86.80	300.77 ± 4.58	2004.02	2	18	88	114.70 ±52.10	202.00 ±91.10	228.77 ± 3.86
p_b^4	b	2003.55	2	1	1	2310.91 ±1378.01	168.00 ±99.80	5517.74 ±27.46	2002.85	2	5	81	469.46 ±221.21	195.00 ±91.60	964.03 ±5.49	2003.25	3	9	7	296.10 ±144.84	209.00 ±102.00	567.61 ±7.20
	w	2009.92	5	19	90	136.76 ±49.51	261.00 ±95.10	210.37 ± 3.42	2003.64	3	15	2	148.46 ±86.03	174.00 ±102.00	340.53 ± 5.61	2003.86	2	20	86	119.16 ±47.15	235.00 ±92.60	203.44 ± 3.10
p_b^5	b	2003.55	2	1	1	2310.91 ±1378.01	168.00 ±100.00	5519.20 ±27.42	2002.69	2	9	91	289.79 ±124.29	223.00 ±96.00	520.32 ±5.14	2003.30	3	8	10	343.78 ±152.71	218.00 ±97.10	631.20 ±5.87
	w	2012.76	46	20	54	165.92 ±35.50	326.00 ±68.90	204.44 ± 3.83	2003.70	3	14	2	170.54 ±87.28	187.00 ±95.80	366.36 ± 5.72	2005.36	13	20	90	129.39 ±31.41	292.00 ±71.60	177.75 ± 2.66
p_b^6	b	2003.55	2	1	1	2310.91 ±1378.01	168.00 ±100.00	5518.13 ±24.01	2002.80	2	7	69	314.25 ±134.66	185.00 ±79.30	680.55 ±5.38	2003.27	2	7	8	384.49 ±201.31	209.00 ±110.00	736.29 ±6.80
	w	2012.07	8	20	90	127.40 ±45.07	270.00 ±95.50	189.73 ± 2.92	2003.62	2	17	1	135.25 ±68.46	181.00 ±91.80	300.09 ± 5.94	2004.19	3	19	94	122.54 ±35.78	249.00 ±72.30	197.43 ± 3.08
p_b^7	b	2003.55	2	1	1	2310.91 ±1378.01	167.00 ±100.00	5519.63 ±27.15	2003.16	2	5	7	456.48 ±255.03	172.00 ±96.00	1059.45 ±6.52	2003.31	2	6	8	374.17 ±210.13	172.00 ±96.90	870.91 ±6.61
	w	2004.65	3	18	3	133.89 ±68.21	193.00 ±97.90	278.72 ± 4.95	2003.86	2	19	79	101.13 ±52.43	189.00 ±98.60	214.40 ± 4.17	2003.91	3	18	99	95.88 ±46.19	180.00 ±87.20	212.91 ± 3.20
p_b^8	b	2003.55	2	1	2	2310.91 ±1378.01	168.00 ±100.00	5507.31 ±27.06	2002.66	3	11	45	229.14 ±88.48	208.00 ±80.60	442.35 ±6.04	**2003.18**	2	3	5	902.52 ±415.33	203.00 ±93.40	1784.74 ±8.61
	w	2011.26	9	20	26	179.76 ±35.63	327.00 ±63.20	220.65 ± 4.38	2003.84	66	18	84	161.46 ±53.90	257.00 ±85.10	252.26 ± 4.27	2005.60	48	20	96	124.02 ±29.97	304.00 ±72.80	163.93 ± 2.52
p_b^9	b	2003.55	2	1	1	2310.91 ±1378.01	167.00 ±100.00	5520.85 ±26.94	2002.69	1	10	77	263.74 ±102.75	227.00 ±88.60	465.00 ±5.36	2003.35	3	10	8	249.93 ±107.84	201.00 ±87.00	499.38 ±5.52
	w	2013.56	71	20	52	156.74 ±36.00	321.00 ±72.70	196.35 ± 3.76	2003.66	3	20	2	118.17 ±58.85	188.00 ±93.90	252.62 ± 4.75	2006.16	42	20	93	123.61 ±30.89	301.00 ±74.90	164.86 ± 2.04

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.29: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra400_16* [Dav06b]. Stopping criterion, $T = 0.4[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\tilde{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\tilde{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\tilde{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	2005.20	2	1	1	2403.47 ±1430.05	186.00 ±111.00	5180.02 ±24.58	2004.75	2	5	15	503.12 ±215.84	211.00 ±90.70	957.32 ±10.07	2004.97	3	4	8	647.42 ±321.19	211.00 ±104.00	1229.29 ±11.70
	w	2006.70	4	19	93	108.02 ±43.93	223.00 ±91.20	194.67 ± 3.16	2005.38	2	19	75	108.48 ±43.40	211.00 ±85.70	206.34 ± 4.49	2005.53	3	18	86	106.38 ±43.13	206.00 ±84.30	207.38 ± 4.36
p_b^1	b	2005.07	9	20	99	123.37 ±24.64	311.00 ±62.20	159.21 ±2.63	2004.39	3	7	60	371.35 ±146.97	233.00 ±92.70	636.18 ±7.14	2004.96	3	3	7	847.70 ±435.32	205.00 ±105.00	1654.80 ±11.19
	w	2012.70	193	20	92	129.17 ±24.08	319.00 ±59.40	162.37 ± 3.02	2006.17	127	15	97	194.74 ±46.91	290.00 ±69.00	269.94 ± 5.51	2012.10	132	19	90	134.88 ±28.61	308.00 ±66.20	175.84 ± 3.58
p_b^2	b	2004.11	7	16	100	143.86 ±37.74	294.00 ±77.00	196.42 ±2.84	**2004.33**	2	8	67	338.18 ±113.60	246.00 ±82.50	550.95 ±7.62	2004.92	2	3	6	793.02 ±414.02	191.00 ±99.30	1662.44 ±10.44
	w	2013.22	163	20	53	146.63 ±27.38	329.00 ±61.00	179.11 ± 3.63	2054.09	218	20	84	184.79 ±35.79	353.00 ±68.30	210.48 ± 4.34	2011.55	162	20	79	135.63 ±24.57	326.00 ±58.20	166.99 ± 3.18
p_b^3	b	2005.08	3	5	2	483.67 ±231.04	193.00 ±92.50	997.98 ±12.27	2005.04	2	3	9	828.81 ±418.45	198.00 ±100.00	1671.81 ±11.10	2005.07	3	4	6	581.58 ±301.99	187.00 ±96.70	1245.69 ±12.24
	w	2006.09	2	20	91	114.46 ±35.39	249.00 ±76.60	185.36 ± 3.95	2005.64	3	20	88	110.64 ±37.81	230.00 ±80.40	192.50 ± 4.45	2005.69	3	19	88	104.44 ±47.86	215.00 ±97.90	195.55 ± 3.94
p_b^4	b	2005.20	2	1	1	2403.47 ±1430.05	186.00 ±110.00	5183.64 ±29.79	2004.59	3	4	85	589.79 ±215.52	213.00 ±77.60	1110.32 ±10.29	2004.93	2	4	7	648.21 ±317.70	211.00 ±104.00	1226.78 ±11.74
	w	2009.61	6	20	76	115.57 ±43.03	246.00 ±91.40	188.47 ± 4.73	2005.59	34	20	90	155.69 ±26.44	316.00 ±54.80	198.20 ± 4.05	2005.64	3	19	89	123.20 ±36.53	259.00 ±77.00	190.75 ± 3.84
p_b^5	b	2005.20	2	1	1	2403.47 ±1430.05	186.00 ±110.00	5189.01 ±26.29	2004.44	3	7	72	358.51 ±129.20	229.00 ±81.70	627.79 ±6.89	2004.96	2	3	13	873.15 ±389.15	218.00 ±97.30	1606.44 ±9.46
	w	2010.65	115	19	78	140.80 ±30.89	317.00 ±68.70	178.24 ± 3.04	2007.13	104	20	91	161.07 ±28.00	321.00 ±54.20	200.92 ± 4.92	2012.95	121	19	94	131.87 ±28.66	313.00 ±67.80	168.90 ± 3.65
p_b^6	b	2005.20	2	1	1	2403.47 ±1430.05	186.00 ±111.00	5183.81 ±24.42	2004.54	2	5	66	503.94 ±200.96	226.00 ±90.10	890.81 ±8.81	**2004.90**	2	6	9	418.26 ±183.15	213.00 ±92.90	789.62 ±10.05
	w	2010.07	7	20	55	122.42 ±42.25	257.00 ±89.50	190.68 ± 4.00	2005.64	18	20	98	154.79 ±26.51	324.00 ±54.40	192.46 ± 4.04	2007.30	106	20	93	124.46 ±31.24	293.00 ±74.10	170.69 ± 2.70
p_b^7	b	2005.20	2	1	1	2403.47 ±1430.05	186.00 ±110.00	5185.06 ±23.18	2004.78	3	5	10	499.16 ±251.99	207.00 ±104.00	965.43 ±10.12	2004.95	2	4	8	588.97 ±313.98	193.00 ±103.00	1223.76 ±10.75
	w	2006.36	4	20	4	115.94 ±50.61	206.00 ±89.40	225.14 ± 5.00	2005.58	3	18	83	112.22 ±42.90	220.00 ±83.50	204.18 ± 4.50	2005.60	3	20	75	94.53 ±36.78	204.00 ±79.10	185.84 ± 3.66
p_b^8	b	***2003.22**	5	20	100	99.98 ±28.94	265.00 ±77.80	152.06 ±2.73	2004.34	3	7	45	406.82 ±142.07	252.00 ±88.30	647.43 ±10.09	2004.92	2	4	6	641.79 ±288.63	210.00 ±94.10	1226.23 ±10.58
	w	2009.81	5	20	9	120.26 ±50.04	222.00 ±92.30	216.98 ± 5.59	2014.34	113	19	96	173.83 ±35.50	324.00 ±66.10	215.48 ± 5.17	2012.55	127	20	92	121.58 ±22.05	322.00 ±57.50	151.65 ± 2.72
p_b^9	b	2005.20	2	1	1	2403.47 ±1430.05	186.00 ±111.00	5189.58 ±24.03	2004.38	2	7	68	380.28 ±123.43	246.00 ±79.90	622.22 ±7.44	2004.98	3	4	7	683.75 ±301.65	226.00 ±99.90	1211.11 ±12.81
	w	2011.41	222	20	71	132.02 ±26.26	317.00 ±63.60	167.61 ± 2.73	2006.69	63	19	87	156.71 ±33.27	301.00 ±65.30	208.22 ± 4.27	2012.29	94	20	78	135.83 ±22.69	336.00 ±56.60	162.20 ± 2.86

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.30: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra450_12* [Dav06b]. Stopping criterion, $T = 0.45[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	2204.17	2	1	1	2578.42 ±1310.21	226.00 ±115.00	5129.73 ±22.48	2203.66	2	4	33	635.73 ±292.54	241.00 ±111.00	1189.82 ±5.15	2203.90	2	8	4	318.39 ±166.71	231.00 ±120.00	622.01 ±5.38
	w	2206.05	4	18	87	118.39 ±52.57	242.00 ±107.00	220.12 ± 2.23	2204.34	2	20	1	125.33 ±55.69	240.00 ±107.00	235.31 ± 3.23	2204.40	3	15	2	162.78 ±77.39	224.00 ±107.00	328.35 ± 3.95
p_b^1	b	2204.17	2	1	1	2578.42 ±1310.21	227.00 ±115.00	5127.13 ±20.95	2203.04	2	8	87	328.64 ±108.77	266.00 ±87.90	557.13 ±4.11	2203.92	3	3	7	874.62 ±390.75	237.00 ±106.00	1658.71 ±10.13
	w	2215.48	14	20	53	141.05 ±49.80	319.00 ±113.00	199.74 ± 2.93	2204.34	2	20	1	125.33 ±55.69	241.00 ±106.00	235.13 ± 3.26	2206.36	52	18	99	144.36 ±37.36	332.00 ±86.10	196.30 ± 2.08
p_b^2	b	2204.17	2	1	1	2578.42 ±1310.21	227.00 ±115.00	5126.25 ±20.87	***2203.00**	2	12	78	215.73 ±78.46	265.00 ±97.10	367.40 ±3.87	2203.93	2	3	7	884.09 ±410.25	241.00 ±112.00	1655.24 ±7.94
	w	2216.37	18	20	40	143.83 ±53.95	319.00 ±119.00	203.42 ± 2.77	2221.28	82	20	97	178.18 ±46.64	382.00 ±99.50	210.85 ± 2.40	2207.17	64	20	73	144.11 ±31.17	350.00 ±75.90	185.96 ± 2.33
p_b^3	b	**2204.10**	3	3	2	920.58 ±434.65	245.00 ±115.00	1693.89 ±7.67	2204.03	2	4	53	621.04 ±283.73	244.00 ±111.00	1147.97 ±5.73	2204.11	2	13	6	215.26 ±91.37	257.00 ±109.00	377.51 ±5.13
	w	2205.15	4	18	100	142.66 ±44.30	299.00 ±92.30	215.11 ± 2.21	2204.56	3	18	94	120.53 ±49.01	248.00 ±101.00	219.19 ± 2.22	2204.61	3	14	59	148.21 ±70.69	215.00 ±102.00	312.20 ± 3.21
p_b^4	b	2204.17	2	1	1	2578.42 ±1310.21	227.00 ±115.00	5126.37 ±20.15	2203.29	2	6	43	417.17 ±178.22	243.00 ±104.00	773.39 ±5.38	2203.89	3	6	9	413.35 ±195.16	228.00 ±108.00	816.72 ±5.92
	w	2212.71	8	20	99	90.25 ±51.04	216.00 ±123.00	188.35 ± 2.15	2204.32	2	19	1	130.90 ±59.15	238.00 ±108.00	248.33 ± 3.36	2204.54	3	20	85	114.63 ±41.70	258.00 ±92.90	200.65 ± 2.25
p_b^5	b	2204.17	2	1	1	2578.42 ±1310.21	226.00 ±115.00	5128.98 ±24.45	2203.08	3	10	74	242.85 ±88.22	248.00 ±90.10	441.93 ±4.67	2203.91	3	4	7	621.85 ±311.11	226.00 ±114.00	1238.27 ±8.91
	w	2215.82	13	20	52	139.40 ±47.79	323.00 ±111.00	194.86 ± 2.15	2204.33	2	20	1	126.72 ±56.74	242.00 ±108.00	235.76 ± 3.25	2207.11	67	19	96	133.57 ±36.34	330.00 ±89.50	183.28 ± 2.32
p_b^6	b	2204.17	2	1	1	2578.42 ±1310.21	226.00 ±115.00	5127.36 ±26.16	2203.22	2	7	91	349.85 ±134.60	253.00 ±97.00	624.23 ±4.09	2203.85	3	3	7	839.32 ±423.21	228.00 ±115.00	1655.48 ±8.30
	w	2214.70	10	20	95	107.41 ±47.15	267.00 ±118.00	181.47 ± 2.16	2204.33	2	18	1	140.45 ±61.08	242.00 ±106.00	261.95 ± 3.09	2204.98	4	18	95	131.90 ±40.06	285.00 ±86.80	209.11 ± 1.83
p_b^7	b	2204.17	2	1	1	2578.42 ±1310.21	226.00 ±115.00	5130.14 ±22.53	2203.69	3	5	8	482.86 ±245.59	223.00 ±113.00	976.84 ±5.81	**2203.81**	3	6	7	414.60 ±192.32	227.00 ±105.00	821.29 ±6.94
	w	2205.60	3	15	4	157.98 ±72.67	229.00 ±105.00	311.67 ± 2.87	2204.50	3	17	70	112.23 ±52.69	217.00 ±102.00	234.18 ± 2.27	2204.60	3	19	96	96.43 ±44.08	215.00 ±97.60	203.11 ± 2.26
p_b^8	b	2204.17	2	1	1	2578.42 ±1310.21	226.00 ±115.00	5130.33 ±20.15	2203.03	2	8	77	297.47 ±120.13	240.00 ±96.00	561.53 ±4.18	2203.86	2	5	8	546.67 ±241.75	251.00 ±111.00	978.55 ±8.68
	w	2213.93	14	20	27	145.76 ±45.71	318.00 ±99.70	207.18 ± 2.65	2205.72	70	19	100	160.75 ±44.72	329.00 ±91.80	220.44 ± 2.90	2207.72	96	19	99	133.04 ±29.01	352.00 ±76.40	170.73 ± 1.77
p_b^9	b	2204.17	2	1	1	2578.42 ±1310.21	226.00 ±115.00	5130.43 ±22.86	2203.12	2	10	90	233.48 ±83.91	245.00 ±88.40	430.04 ±3.96	2203.93	3	8	9	339.61 ±148.75	255.00 ±112.00	599.03 ±5.62
	w	2216.19	13	20	52	138.02 ±40.69	333.00 ±97.90	187.44 ± 1.95	2204.34	2	20	1	125.40 ±55.72	242.00 ±107.00	234.38 ± 2.73	2206.95	65	19	89	131.08 ±34.17	333.00 ±87.70	177.55 ± 1.86

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.31: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra450_16* [Dav06b]. Stopping criterion, $T = 0.45[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	2206.51	4	1	1	2428.32 ±1109.50	226.00 ±103.00	4830.56 ±19.22	2205.83	3	3	11	966.76 ±362.07	282.00 ±106.00	1545.51 ±8.74	2205.94	4	3	14	857.14 ±366.10	251.00 ±107.00	1539.72 ±11.27
	w	2208.79	4	19	85	128.57 ±37.31	304.00 ±87.80	191.03 ± 2.65	2206.69	4	18	1	150.26 ±49.38	285.00 ±92.90	238.44 ± 3.98	2206.66	4	18	1	153.58 ±51.87	274.00 ±93.40	252.63 ± 4.34
p_b^1	b	2206.51	4	1	1	2428.32 ±1109.50	226.00 ±103.00	4829.45 ±23.34	2205.26	3	7	70	376.23 ±113.71	285.00 ±86.40	593.23 ±5.09	2206.03	3	3	11	882.68 ±338.61	259.00 ±99.40	1534.38 ±10.96
	w	2221.58	10	20	40	164.01 ±22.64	386.00 ±53.30	192.16 ± 2.04	2210.92	161	20	69	165.52 ±24.93	385.00 ±58.00	194.34 ± 2.85	2210.08	192	17	80	164.09 ±25.68	363.00 ±57.00	204.33 ± 2.54
p_b^2	b	2206.51	4	1	1	2428.32 ±1109.50	226.00 ±103.00	4830.92 ±22.77	***2205.10**	3	6	71	453.16 ±144.52	290.00 ±92.40	704.03 ±6.46	2205.97	2	3	11	866.15 ±332.01	256.00 ±98.30	1525.00 ±10.34
	w	2223.58	21	20	39	151.38 ±27.16	371.00 ±65.80	184.02 ± 3.50	2245.45	269	18	86	187.74 ±25.34	393.00 ±52.80	215.64 ± 2.97	2210.58	148	18	86	146.43 ±25.57	359.00 ±63.00	184.04 ± 2.46
p_b^3	b	**2206.31**	4	2	33	1332.31 ±527.97	265.00 ±105.00	2265.88 ±10.24	2206.23	3	4	29	648.80 ±241.02	265.00 ±98.80	1101.95 ±7.69	2206.30	3	2	31	1376.24 ±527.73	274.00 ±105.00	2261.66 ±14.84
	w	2207.62	3	20	99	132.87 ±28.06	330.00 ±69.60	181.90 ± 2.11	2207.01	3	20	99	122.46 ±29.42	309.00 ±73.90	179.11 ± 2.12	2206.95	2	19	83	123.20 ±41.84	278.00 ±94.90	200.17 ± 2.94
p_b^4	b	2206.51	4	1	1	2428.32 ±1109.50	227.00 ±103.00	4829.44 ±21.80	2205.48	3	4	61	655.67 ±218.86	276.00 ±91.60	1070.78 ±6.57	2205.96	2	3	25	885.38 ±342.05	267.00 ±103.00	1493.06 ±9.66
	w	2217.50	8	20	100	119.55 ±31.54	312.00 ±83.20	173.38 ± 3.84	2207.95	71	18	79	179.05 ±29.00	388.00 ±62.70	208.37 ± 3.06	2206.87	3	19	100	135.21 ±32.06	319.00 ±77.10	190.65 ± 2.85
p_b^5	b	2206.49	4	1	1	2470.68 ±1120.73	231.00 ±105.00	4825.34 ±24.76	2205.27	2	5	74	529.88 ±177.86	283.00 ±94.70	843.52 ±6.20	2206.09	4	5	7	571.70 ±189.07	281.00 ±93.00	917.95 ±8.30
	w	2222.56	286	20	100	133.03 ±19.43	385.00 ±56.80	156.12 ± 3.01	2212.38	140	19	85	175.56 ±22.62	396.00 ±50.40	200.49 ± 2.74	2210.81	178	19	96	142.17 ±21.60	377.00 ±57.00	170.48 ± 1.73
p_b^6	b	2206.51	4	1	1	2428.32 ±1109.50	226.00 ±104.00	4833.92 ±20.44	2205.43	2	5	82	508.45 ±165.79	279.00 ±90.80	822.11 ±5.76	**2205.90**	4	3	10	907.32 ±341.83	267.00 ±100.00	1533.08 ±10.85
	w	2220.34	9	20	97	123.94 ±29.50	342.00 ±81.20	163.74 ± 1.93	2208.92	38	20	87	164.96 ±21.52	391.00 ±51.50	190.54 ± 2.61	2208.03	81	20	84	136.52 ±25.95	348.00 ±65.80	177.12 ± 2.68
p_b^7	b	2206.51	4	1	1	2428.32 ±1109.50	227.00 ±104.00	4830.20 ±22.21	2205.89	2	4	9	641.35 ±258.94	254.00 ±103.00	1136.39 ±8.91	2205.94	4	5	11	535.70 ±216.14	264.00 ±107.00	912.50 ±8.19
	w	2208.30	4	17	5	158.12 ±54.72	288.00 ±99.40	248.24 ± 3.34	2206.86	4	20	84	108.75 ±34.96	277.00 ±90.10	176.76 ± 2.81	2206.92	3	19	100	114.92 ±38.46	278.00 ±94.40	185.95 ± 2.57
p_b^8	b	2206.51	4	1	1	2428.32 ±1109.50	227.00 ±104.00	4825.58 ±23.02	2205.19	3	8	74	341.13 ±89.34	299.00 ±78.00	514.91 ±4.49	2205.97	3	3	10	903.97 ±356.80	266.00 ±105.00	1528.83 ±9.07
	w	2224.28	389	20	88	139.93 ±16.88	405.00 ±47.60	155.99 ± 2.34	2220.57	295	19	74	185.34 ±22.27	410.00 ±48.30	204.35 ± 2.72	2211.94	57	20	96	134.97 ±19.95	392.00 ±58.70	155.36 ± 2.06
p_b^9	b	2206.51	4	1	1	2428.32 ±1109.50	226.00 ±104.00	4835.42 ±21.79	2205.28	3	6	69	434.80 ±134.61	284.00 ±88.60	689.40 ±6.06	2206.06	3	2	12	1435.10 ±518.96	284.00 ±103.00	2271.47 ±12.71
	w	2222.59	16	20	44	151.45 ±20.27	384.00 ±50.90	178.06 ± 3.20	2211.14	73	20	74	172.02 ±24.56	402.00 ±56.70	193.14 ± 3.19	2210.91	173	19	80	143.07 ±20.60	377.00 ±54.50	171.39 ± 2.19

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.32: Best and worst average solutions found by corresponding BCOs for problem instance *logra500_12* [Dav06b]. Stopping criterion, $T = 0.5[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	2403.26	2	1	1	2016.65 ±1191.67	210.00 ±124.00	4800.91 ±21.35	2402.86	2	8	8	280.44 ±140.37	248.00 ±124.00	565.73 ±5.01	2403.07	2	6	7	339.54 ±184.41	223.00 ±121.00	762.32 ±5.18
	w	2404.52	3	20	91	80.90 ±45.24	212.00 ±119.00	191.36 ± 2.70	2403.37	2	17	2	120.17 ±64.32	229.00 ±122.00	262.89 ± 2.94	2403.49	2	15	70	110.24 ±58.43	209.00 ±111.00	264.11 ± 3.43
p_b^1	b	2403.03	5	18	99	111.50 ±41.28	295.00 ±109.00	188.95 ±2.45	2402.53	1	9	82	229.62 ±110.41	244.00 ±117.00	471.52 ±4.75	2403.15	2	3	6	730.07 ±396.05	235.00 ±127.00	1555.90 ±8.82
	w	2408.25	6	20	13	81.39 ±60.56	193.00 ±144.00	211.20 ± 2.43	2403.38	2	17	2	117.96 ±63.16	225.00 ±121.00	262.25 ± 3.38	2405.22	8	20	95	135.24 ±22.21	400.00 ±65.70	169.59 ± 2.29
p_b^2	b	2402.35	3	17	100	124.14 ±40.45	323.00 ±105.00	192.79 ±2.79	2402.46	1	12	56	178.83 ±74.59	250.00 ±104.00	357.07 ±4.30	2403.15	2	3	3	676.41 ±405.55	215.00 ±129.00	1573.38 ±7.74
	w	2409.41	9	20	13	78.69 ±62.00	188.00 ±148.00	210.27 ± 3.04	2404.44	47	19	87	125.46 ±45.53	291.00 ±106.00	215.97 ± 2.77	2406.00	42	20	99	140.48 ±22.86	430.00 ±69.80	164.01 ± 2.22
p_b^3	b	2403.16	2	5	15	468.18 ±248.51	258.00 ±136.00	910.78 ±5.88	2403.16	2	4	35	502.72 ±299.62	225.00 ±135.00	1115.89 ±6.85	2403.13	2	3	10	722.00 ±422.94	232.00 ±136.00	1554.89 ±7.88
	w	2403.94	2	20	89	111.78 ±41.38	295.00 ±109.00	190.60 ± 2.45	2403.61	2	17	92	95.80 ±44.66	211.00 ±99.30	227.01 ± 2.65	2403.70	2	18	94	112.10 ±47.33	268.00 ±112.00	210.02 ± 2.77
p_b^4	b	2403.25	2	1	1	2061.36 ±1212.26	215.00 ±126.00	4801.86 ±25.10	2402.66	2	6	76	332.44 ±160.27	233.00 ±112.00	715.33 ±4.97	2403.09	2	5	9	412.56 ±234.33	226.00 ±128.00	914.51 ±6.17
	w	2407.78	5	20	66	91.93 ±53.93	237.00 ±139.00	194.49 ± 2.85	2403.39	2	17	2	116.14 ±61.59	222.00 ±118.00	262.35 ± 3.48	2403.72	2	19	91	105.25 ±40.97	270.00 ±105.00	195.77 ± 2.42
p_b^5	b	2403.24	5	19	100	105.91 ±35.86	305.00 ±104.00	174.39 ±2.01	2402.52	3	7	44	321.34 ±145.87	256.00 ±117.00	629.04 ±5.43	2403.14	2	5	9	385.58 ±214.42	212.00 ±118.00	908.85 ±5.99
	w	2408.48	7	20	27	93.28 ±60.26	233.00 ±151.00	200.97 ± 2.52	2403.37	2	19	2	109.75 ±58.36	237.00 ±126.00	233.41 ± 3.68	2405.36	30	20	98	131.76 ±23.89	403.00 ±72.80	164.00 ± 1.84
p_b^6	b	2403.26	2	1	1	2016.65 ±1191.67	210.00 ±124.00	4799.78 ±17.71	2402.61	3	5	93	397.49 ±220.23	235.00 ±130.00	848.47 ±6.64	2403.08	2	10	5	207.71 ±110.07	233.00 ±124.00	447.44 ±4.33
	w	2408.02	8	20	40	76.72 ±57.23	193.00 ±144.00	198.25 ± 2.98	2403.32	2	20	2	100.07 ±54.90	225.00 ±124.00	222.36 ± 3.79	2404.08	2	19	86	121.04 ±33.00	321.00 ±87.40	189.18 ± 2.29
p_b^7	b	2403.26	2	1	1	2016.65 ±1191.67	210.00 ±124.00	4797.48 ±15.94	2402.90	2	5	9	407.18 ±234.49	222.00 ±128.00	917.45 ±5.85	**2403.05**	2	3	8	669.31 ±417.11	216.00 ±135.00	1548.53 ±8.04
	w	2404.29	2	13	5	154.60 ±84.46	226.00 ±125.00	340.64 ± 4.14	2403.50	2	15	90	105.53 ±59.56	212.00 ±119.00	249.48 ± 2.56	2403.61	3	14	96	109.46 ±66.87	209.00 ±128.00	263.12 ± 3.74
p_b^8	b	***2401.71**	2	20	96	109.88 ±35.86	339.00 ±110.00	162.56 ±2.23	**2402.43**	1	12	67	186.48 ±80.66	264.00 ±113.00	353.82 ±4.59	2403.11	2	2	6	1115.93 ±618.48	238.00 ±132.00	2344.28 ±10.03
	w	2408.00	7	19	13	97.77 ±65.48	223.00 ±150.00	220.35 ± 2.88	2403.51	76	20	95	115.96 ±37.74	285.00 ±93.00	203.66 ± 2.81	2406.99	54	20	94	134.64 ±17.00	435.00 ±54.80	155.50 ± 1.93
p_b^9	b	2403.21	4	20	100	105.52 ±32.39	334.00 ±103.00	158.51 ±2.47	2402.53	1	8	69	267.38 ±123.89	252.00 ±117.00	530.45 ±4.44	2403.17	2	4	9	507.56 ±285.03	224.00 ±126.00	1132.82 ±6.14
	w	2408.33	8	20	26	83.57 ±55.28	212.00 ±140.00	198.89 ± 2.81	2403.37	2	13	2	146.94 ±88.89	212.00 ±130.00	346.72 ± 4.48	2405.37	10	20	95	128.35 ±18.33	412.00 ±59.00	156.57 ± 2.01

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

Table B.33: Best and worst average solutions found by corresponding BCOs for problem instance *Iogra500_16* [Dav06b]. Stopping criterion, $T = 0.5[s]$.

		min, f_b^1							max, f_b^1							max, f_b^2						
p_b		$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$	$\bar{y}$	Δy	B	NC	$\overline{n_{it}} \pm s$	$\bar{t} \pm s$ $[10^{-3}]$	$\overline{N_{it}} \pm s$
p_b^0	b	2406.26	3	1	1	2409.20 ±1139.88	265.00 ±126.00	4543.11 ±17.75	2405.81	3	3	29	813.86 ±329.98	286.00 ±117.00	1422.02 ±9.63	2406.02	3	4	24	640.54 ±241.37	306.00 ±115.00	1047.35 ±7.64
	w	2408.25	4	18	93	115.92 ±44.27	299.00 ±114.00	195.09 ± 3.43	2406.58	3	18	75	113.89 ±41.05	284.00 ±103.00	201.36 ± 3.19	2406.74	4	15	91	137.05 ±47.11	294.00 ±100.00	233.96 ± 3.68
p_b^1	b	2406.26	3	1	1	2409.20 ±1139.88	265.00 ±126.00	4542.22 ±21.31	2405.29	3	6	64	439.96 ±148.94	327.00 ±111.00	672.82 ±7.42	2406.16	3	4	6	604.58 ±249.22	281.00 ±117.00	1077.21 ±9.98
	w	2413.89	9	20	21	100.89 ±54.63	268.00 ±145.00	188.55 ± 2.87	2418.35	89	19	98	181.31 ±16.42	469.00 ±41.20	193.88 ± 4.03	2465.39	181	20	100	157.70 ±8.81	493.00 ±27.00	160.66 ± 3.74
p_b^2	b	2406.26	3	1	1	2409.20 ±1139.88	265.00 ±125.00	4543.31 ±20.83	***2405.25**	2	7	56	399.27 ±110.22	345.00 ±95.40	578.66 ±6.50	2406.15	4	4	7	645.23 ±230.20	302.00 ±108.00	1069.85 ±8.60
	w	2416.59	174	19	95	133.82 ±22.20	425.00 ±69.70	158.08 ± 2.33	2541.25	296	20	88	185.14 ±10.10	492.00 ±24.60	188.78 ± 3.35	2477.56	262	20	99	150.88 ±6.32	497.00 ±19.90	152.56 ± 2.00
p_b^3	b	2406.22	3	4	2	600.63 ±254.09	273.00 ±115.00	1101.31 ±8.79	2406.19	3	2	11	1160.05 ±539.00	261.00 ±122.00	2222.37 ±9.21	2406.22	4	4	3	611.00 ±262.22	278.00 ±119.00	1100.35 ±7.80
	w	2407.40	3	20	99	121.96 ±32.52	355.00 ±93.70	172.92 ± 2.62	2407.02	4	20	89	117.22 ±28.19	334.00 ±78.90	175.83 ± 2.64	2407.09	3	20	98	115.02 ±26.47	333.00 ±76.30	173.31 ± 2.78
p_b^4	b	2406.26	3	1	1	2409.20 ±1139.88	265.00 ±125.00	4545.74 ±18.40	2405.54	3	4	85	604.75 ±208.56	302.00 ±104.00	1001.05 ±8.04	2406.03	4	2	6	1252.22 ±589.98	282.00 ±133.00	2220.18 ±11.32
	w	2412.92	9	20	93	99.00 ±45.39	285.00 ±130.00	174.49 ± 2.69	2409.79	40	20	94	167.77 ±17.72	457.00 ±48.40	184.43 ± 2.88	2407.55	3	19	98	146.24 ±21.05	410.00 ±58.40	178.97 ± 2.96
p_b^5	b	2406.26	3	1	1	2409.20 ±1139.88	265.00 ±126.00	4546.06 ±20.43	2405.35	3	6	71	429.93 ±129.90	322.00 ±97.10	666.75 ±7.25	2406.15	3	3	5	804.15 ±351.99	274.00 ±120.00	1467.43 ±8.64
	w	2413.67	10	20	25	97.26 ±51.51	263.00 ±138.00	185.81 ± 2.96	2418.48	83	20	97	172.80 ±20.50	470.00 ±54.10	184.52 ± 3.21	2467.31	188	20	97	151.48 ±6.32	496.00 ±19.00	153.25 ± 2.28
p_b^6	b	2406.26	3	1	1	2409.20 ±1139.88	265.00 ±126.00	4545.08 ±21.58	2405.50	3	5	90	500.78 ±161.57	319.00 ±102.00	786.45 ±7.99	2406.10	4	3	5	834.69 ±337.19	286.00 ±116.00	1461.01 ±10.91
	w	2413.38	9	20	57	98.61 ±41.37	276.00 ±116.00	179.28 ± 3.56	2412.28	53	20	100	164.94 ±14.69	466.00 ±41.50	177.72 ± 2.93	2413.57	68	20	96	151.80 ±14.76	460.00 ±44.50	165.75 ± 2.65
p_b^7	b	2406.26	3	1	1	2409.20 ±1139.88	266.00 ±126.00	4537.74 ±19.61	2405.89	3	4	8	579.75 ±258.06	269.00 ±120.00	1078.19 ±9.46	**2406.00**	3	3	10	794.98 ±357.75	276.00 ±124.00	1444.56 ±11.10
	w	2407.88	4	20	7	120.98 ±43.01	305.00 ±108.00	199.05 ± 3.68	2406.74	3	17	94	122.61 ±38.60	308.00 ±97.90	199.52 ± 2.68	2406.81	3	16	80	124.09 ±49.03	288.00 ±112.00	216.11 ± 3.66
p_b^8	b	**2406.11**	8	15	98	150.68 ±35.26	375.00 ±86.60	201.24 ±2.78	2405.27	3	6	49	438.89 ±134.62	323.00 ±98.80	680.12 ±8.18	2406.13	3	2	6	1151.34 ±506.62	261.00 ±115.00	2211.27 ±9.10
	w	2413.18	166	19	87	129.02 ±24.78	405.00 ±78.70	159.77 ± 2.55	2441.56	192	19	94	183.98 ±17.90	469.00 ±44.60	196.63 ± 3.61	2499.37	216	20	99	140.40 ±6.43	497.00 ±20.50	141.86 ± 2.16
p_b^9	b	2406.26	3	1	1	2409.20 ±1139.88	265.00 ±126.00	4545.61 ±17.74	2405.38	3	7	67	378.58 ±102.32	336.00 ±90.40	565.26 ±5.85	2406.18	4	2	9	1249.82 ±500.35	288.00 ±116.00	2172.22 ±8.43
	w	2413.88	7	20	22	83.43 ±51.81	226.00 ±141.00	185.27 ± 3.38	2419.38	112	20	99	175.25 ±11.37	482.00 ±30.70	182.51 ± 3.13	2477.83	192	20	94	143.97 ±5.96	497.00 ±20.10	145.50 ± 2.35

[1] **b** = best BCO configuration,
w = worst BCO configuration.
[2] Overall best solution; Best within its group (overall second best); Best within its group (overall third best);

B.3 Bar-chart of instances

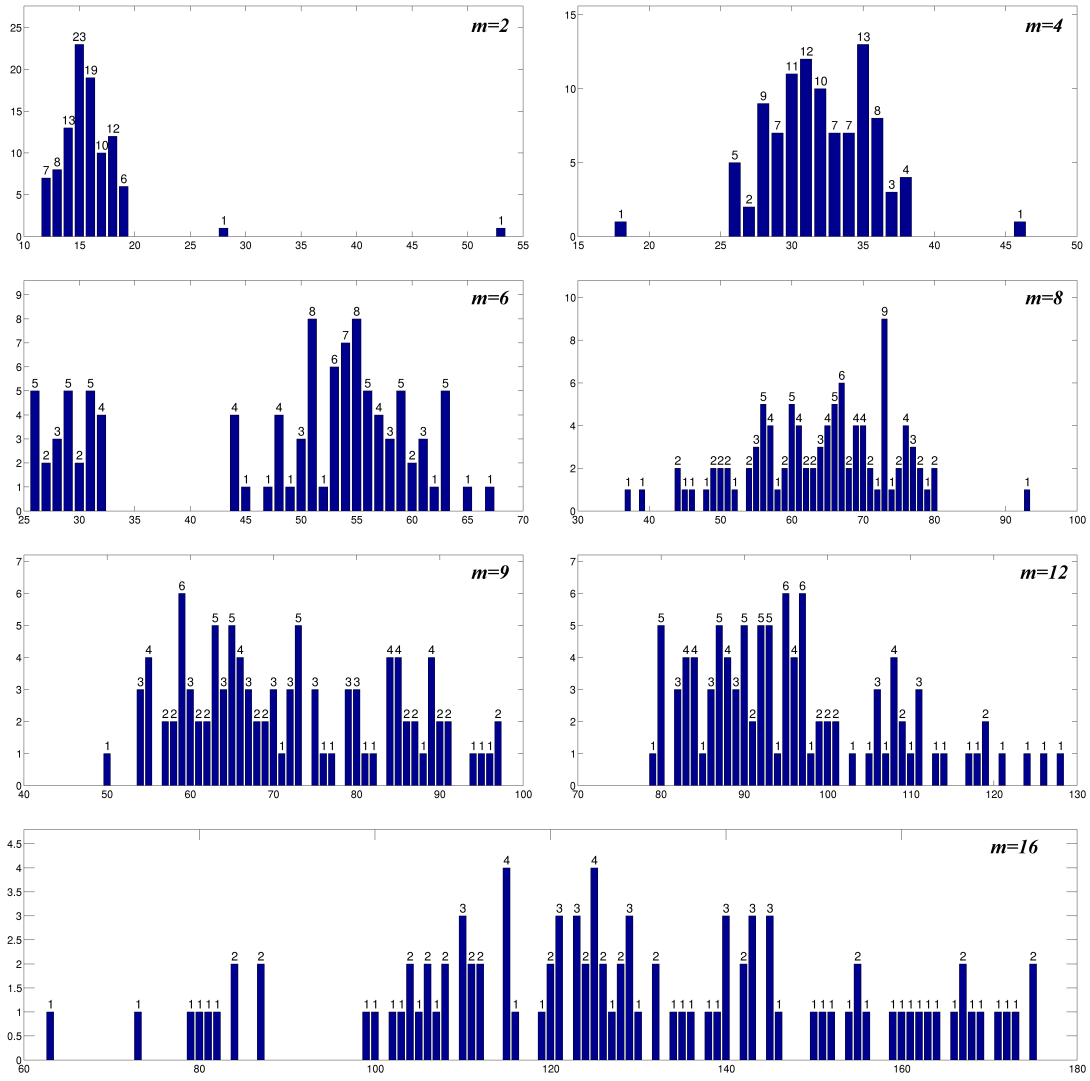


Figure B.8: Bar chart for Iogra100

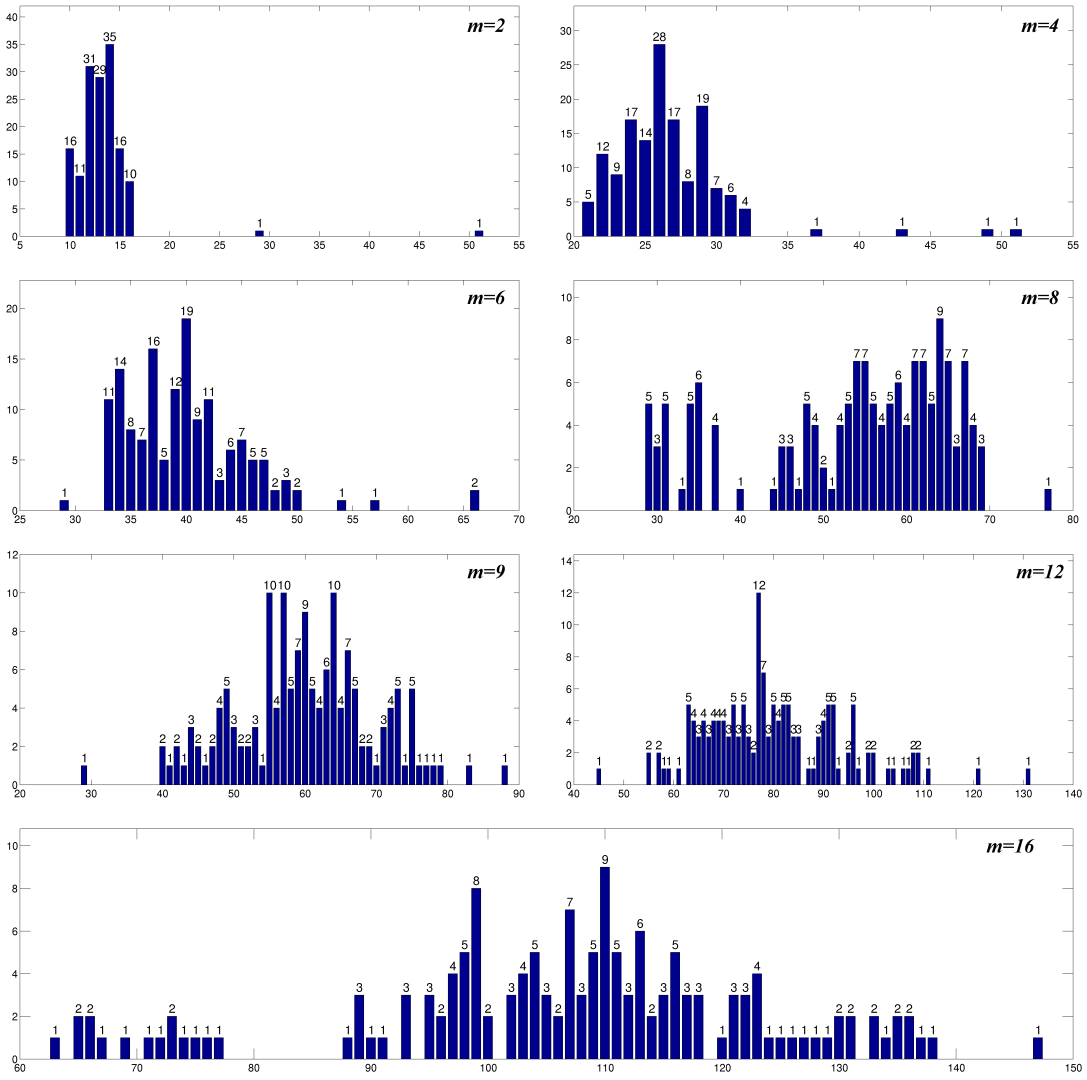


Figure B.9: Bar chart for Iogra150

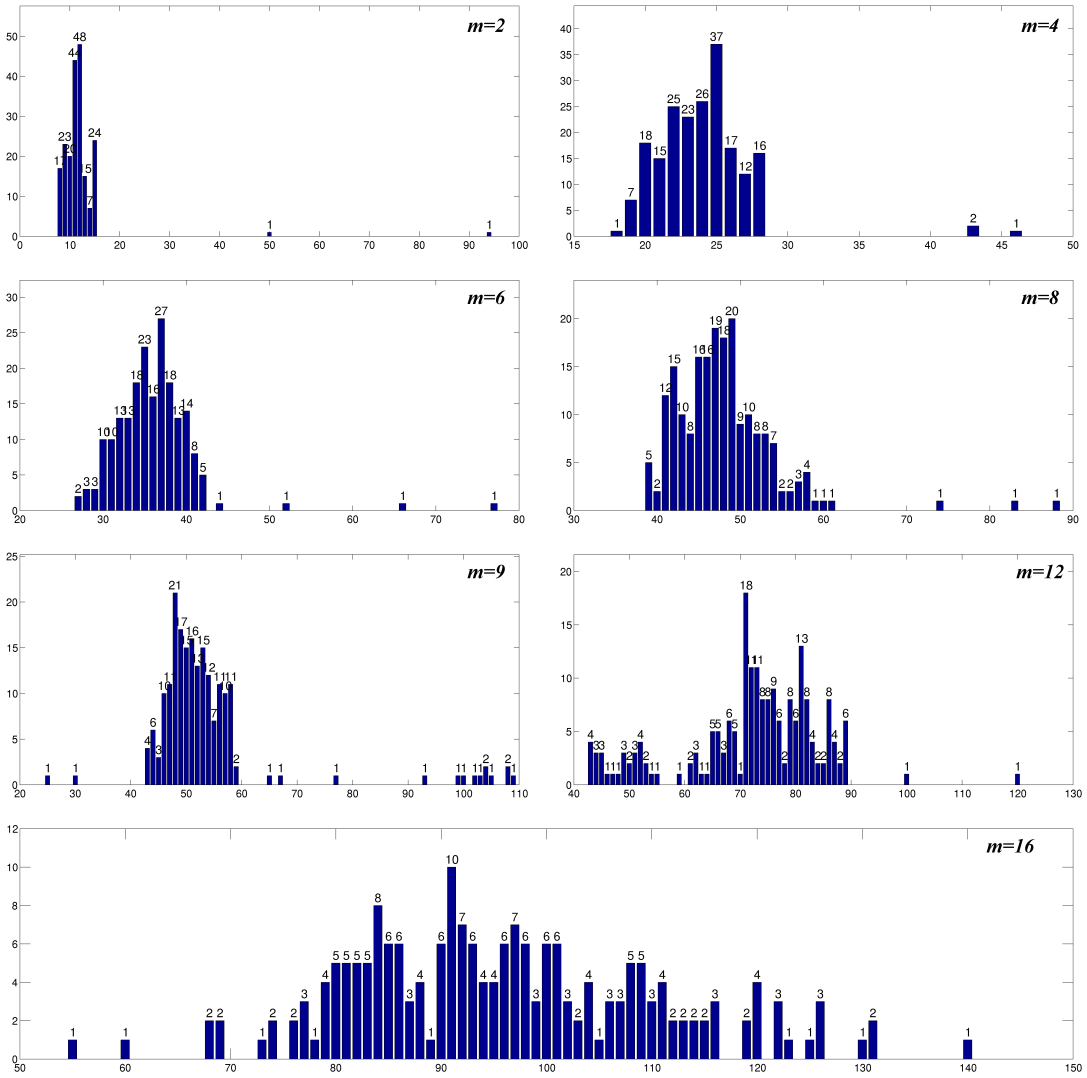


Figure B.10: Bar chart for Iogra200

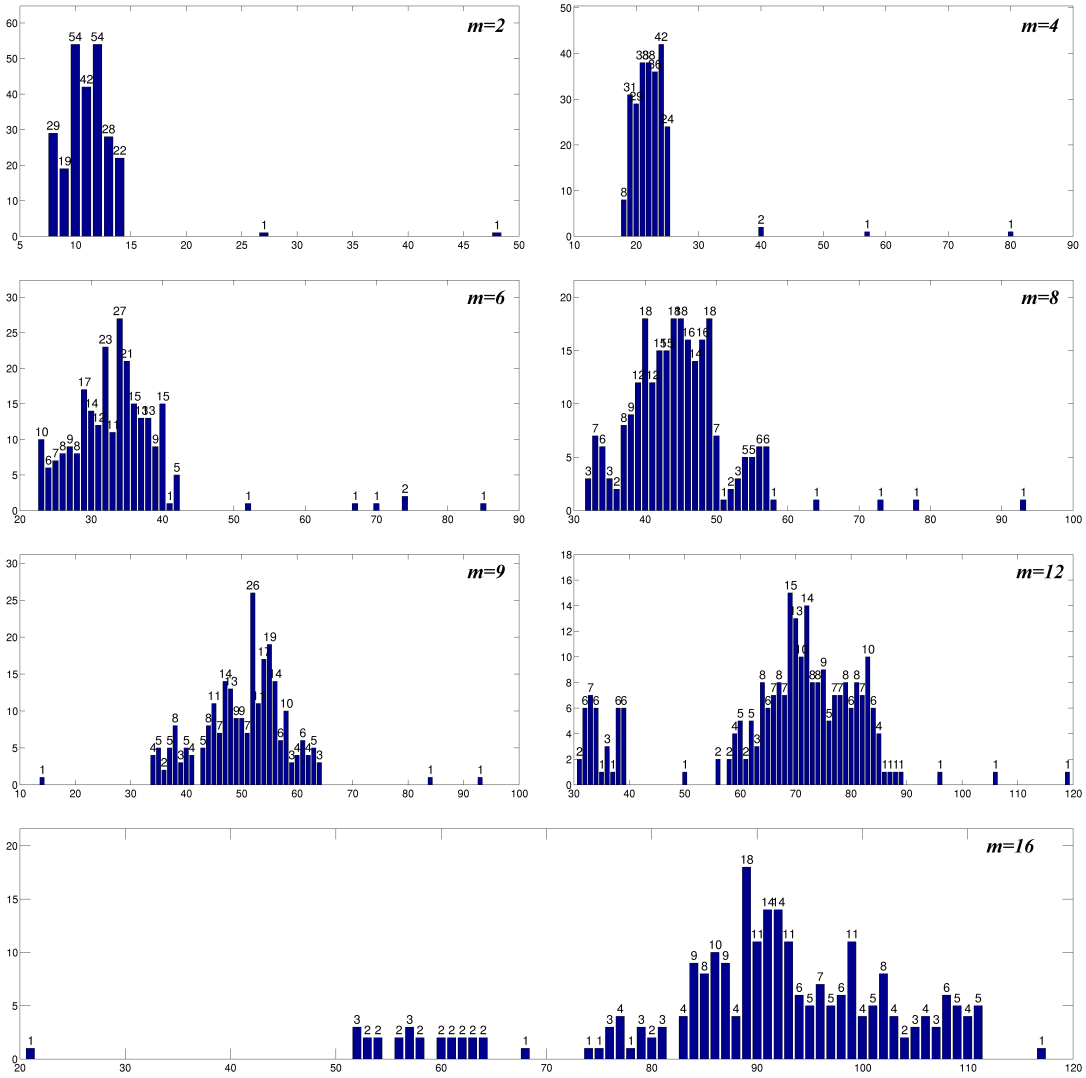


Figure B.11: Bar chart for Iogra250

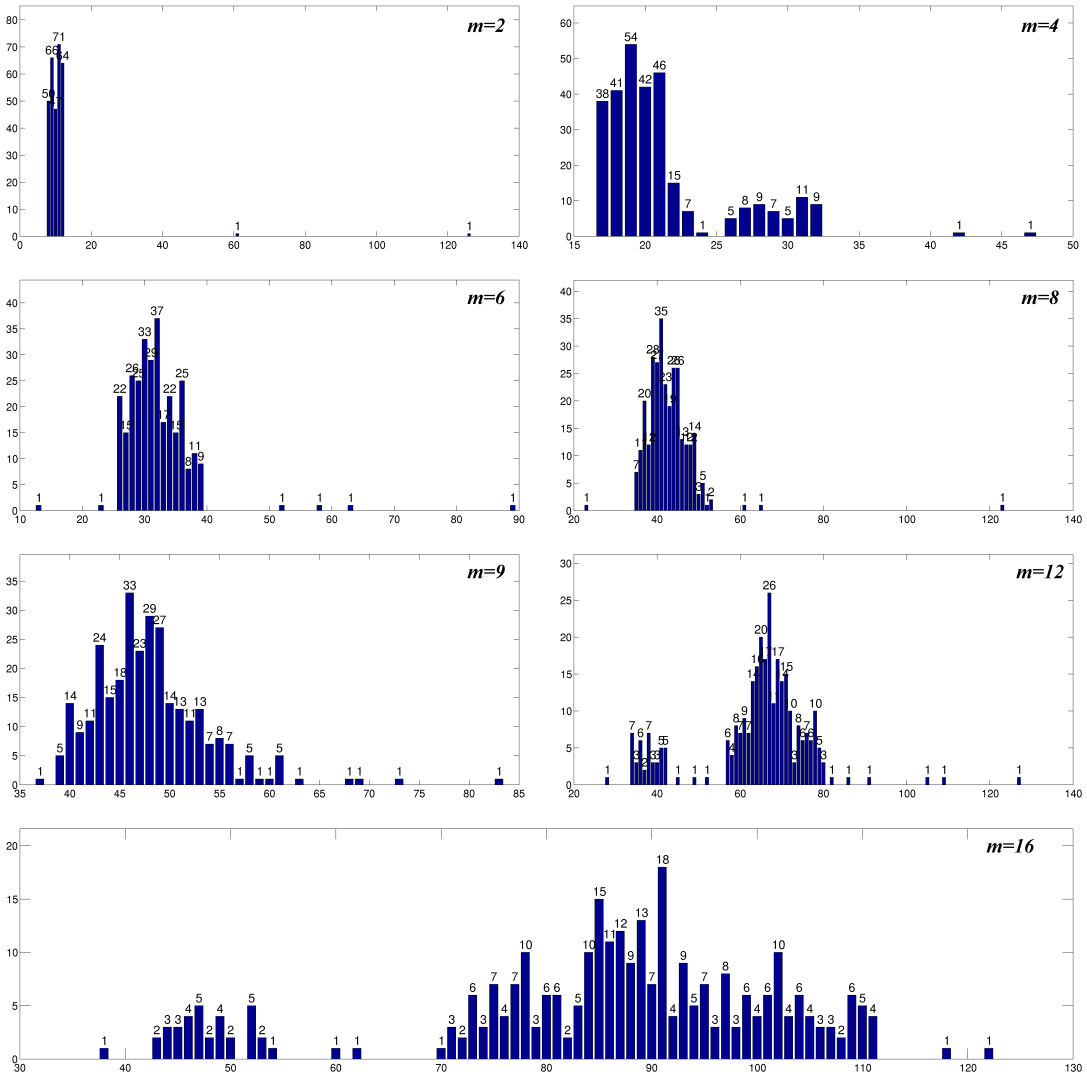


Figure B.12: Bar chart for Iogra300

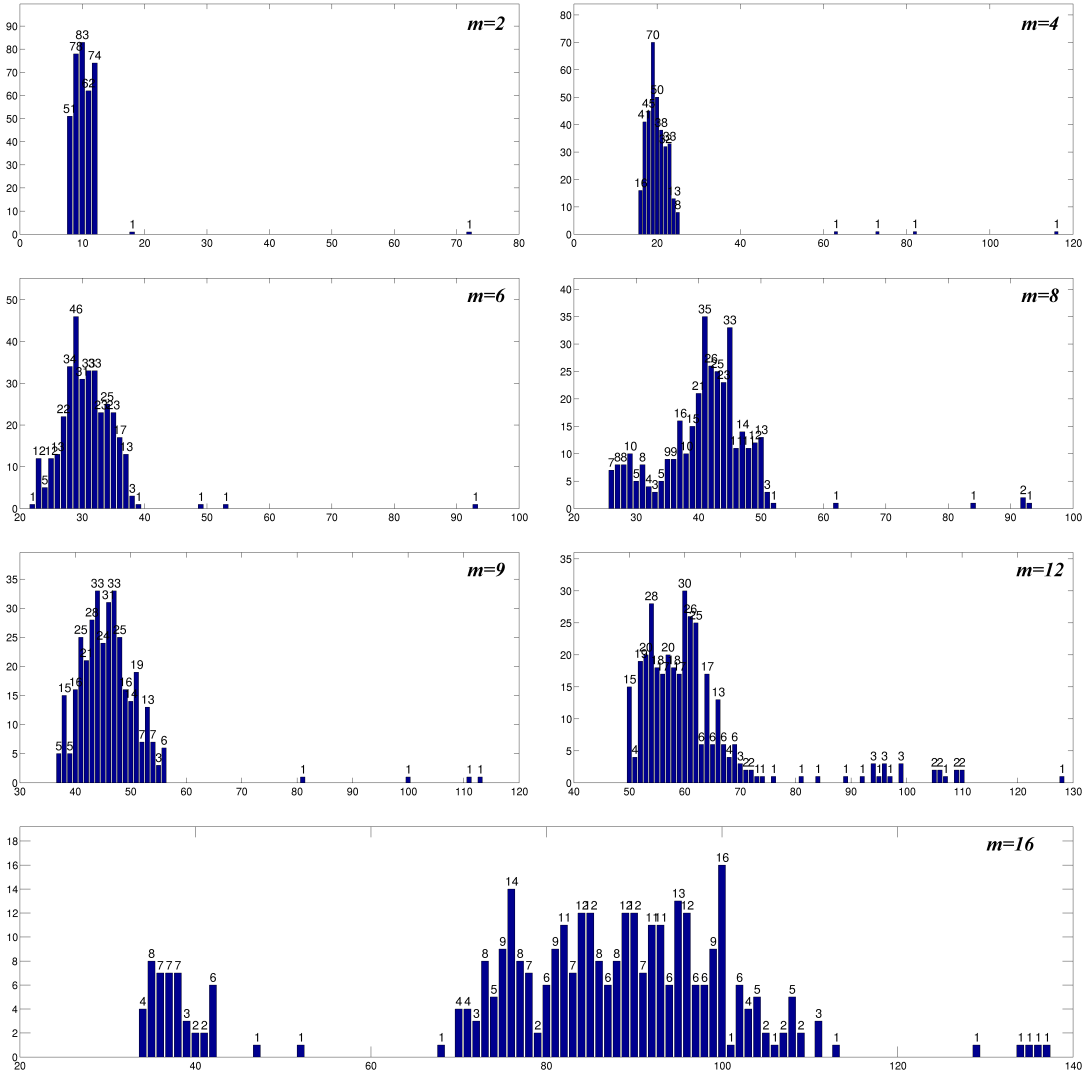


Figure B.13: Bar chart for Iogra350

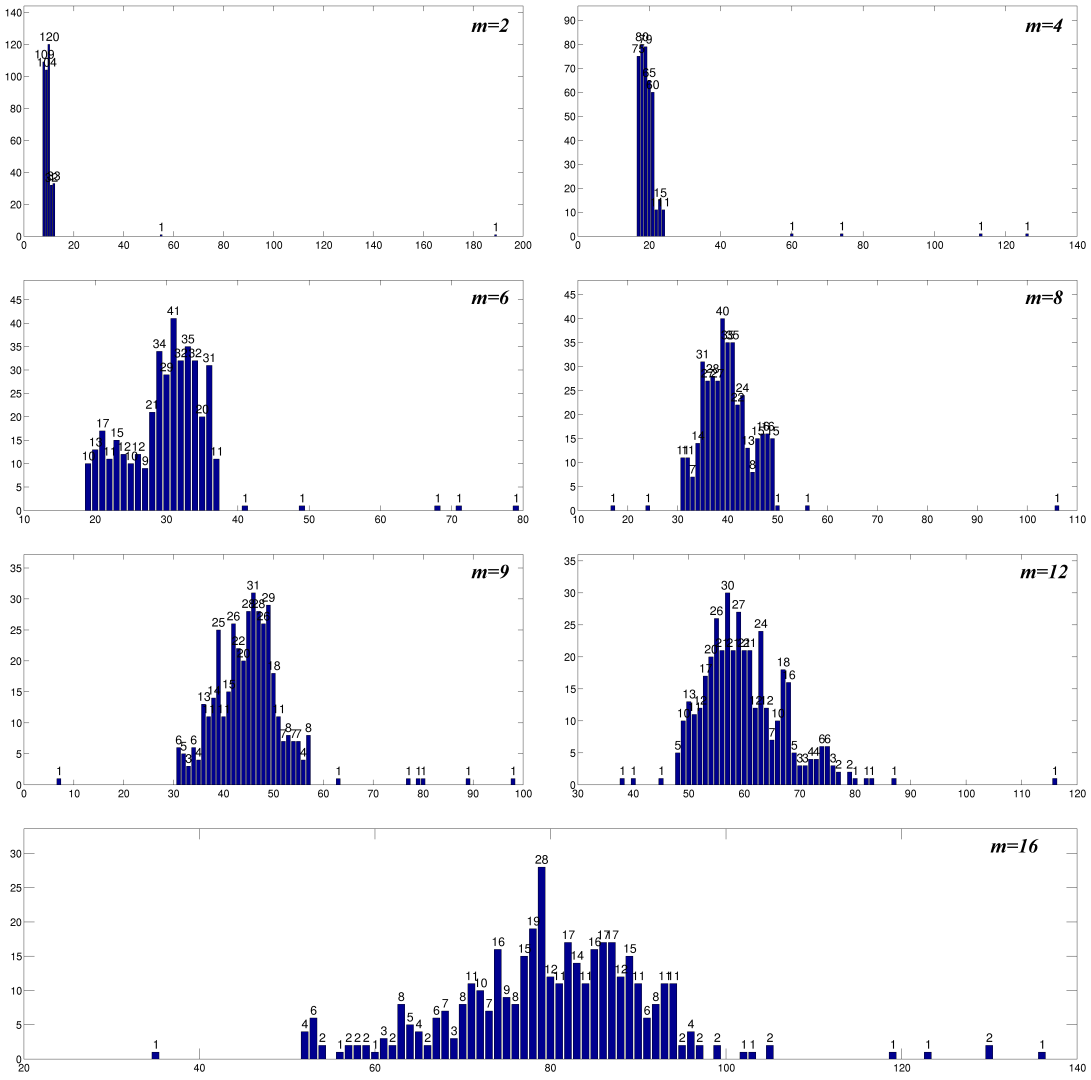


Figure B.14: Bar chart for Iogra400

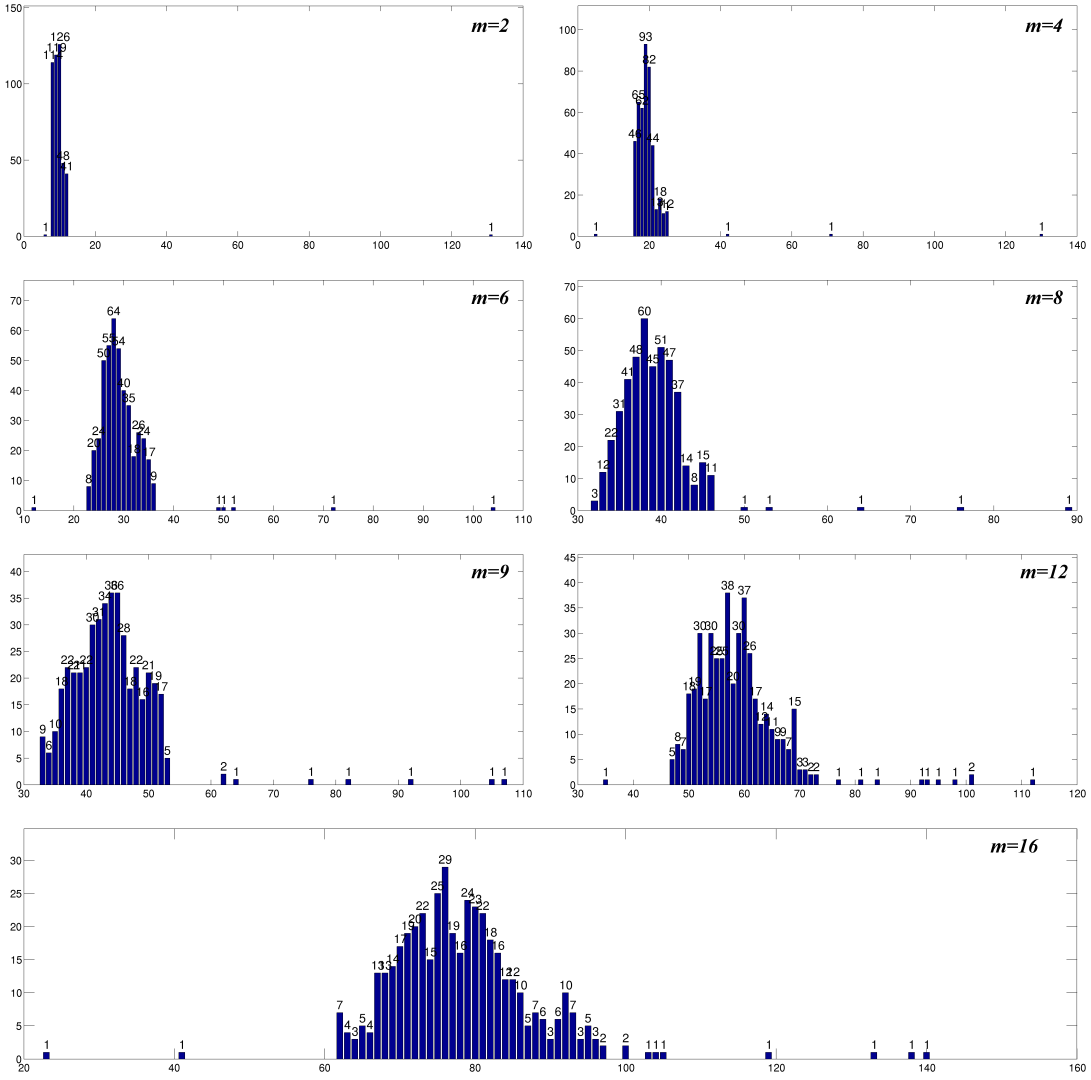


Figure B.15: Bar chart for Iogra450

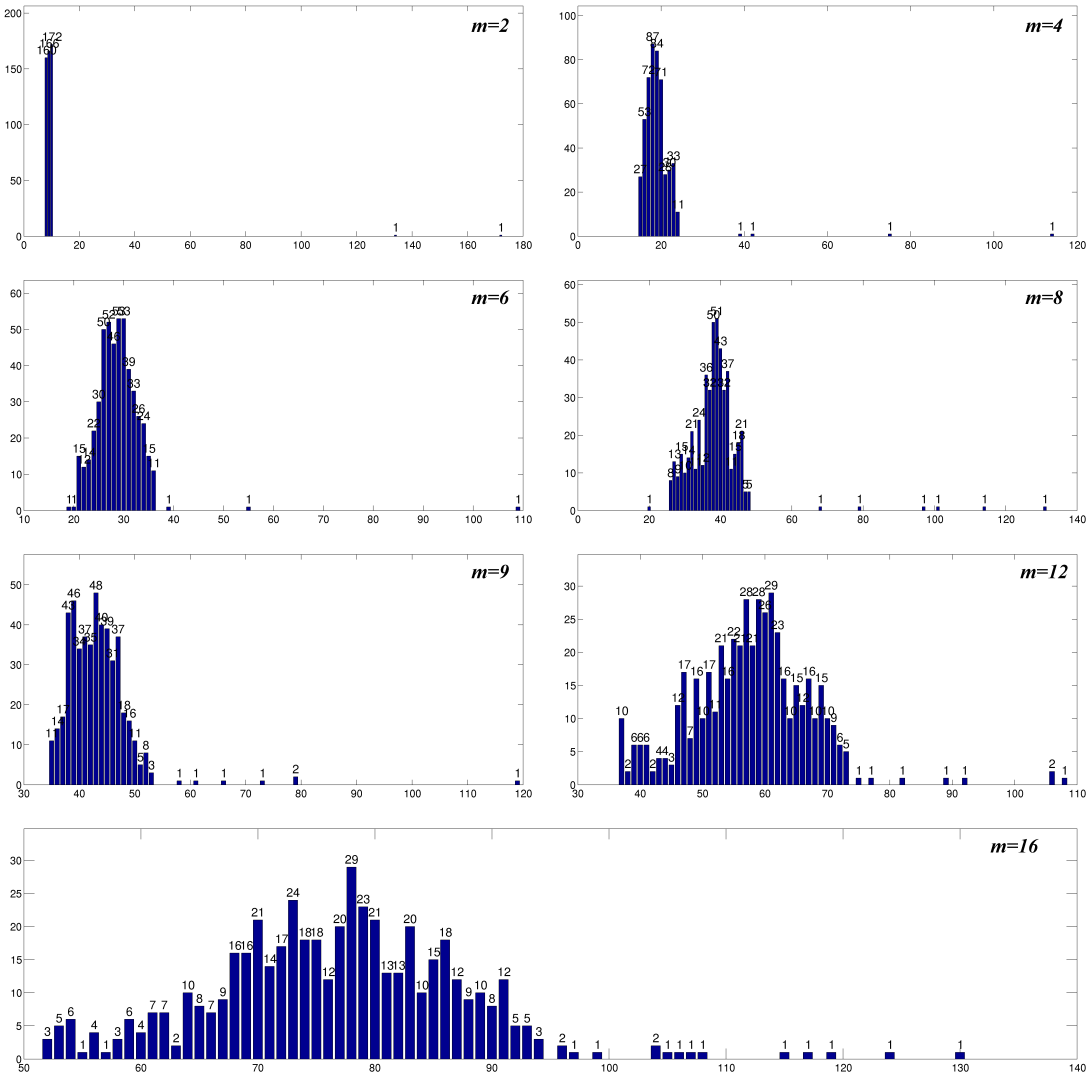


Figure B.16: Bar chart for Iogra500