SENSITIVITY ANALYSIS OF THE BEE COLONY OPTIMIZATION ALGORITHM

Tatjana Jakšić Krüger
Mathematical Institute of Serbian Academy of Sciences and Arts, Belgrade, Serbia
Faculty of Technical Sciences, Novi Sad, Serbia
tatjana@mi.sanu.ac.rs

Tatjana Davidović
Mathematical Institute of Serbian Academy of Sciences and Arts, Belgrade, Serbia
tanjad@mi.sanu.ac.rs

Abstract Bee Colony Optimization (BCO) is a nature-inspired population-based meta-heuristic method that belongs to the class of Swarm intelligence algorithms. BCO was proposed by Lučić and Teodorović, who were among the first to use the basic principles of collective bee intelligence in dealing with optimization problems. Designing a BCO method in principle includes choosing a procedure for constructive/improvement moves, an evaluation function and setting BCO parameters to a suitable values. Topic of this work is addressing the influence of right choice of BCO underlying procedures, such as the choice of loyalty functions, and influence of parameter variations on algorithm performance, by means of visual inspection. Analyses were conducted for simple variant of scheduling problem. Also, to achieve good alternatives for reported solutions, new evaluation methods for scheduling problem are presented.

Keywords: Empirical analysis, Meta-heuristics, Parameter tuning, Swarm intelligence.

1. Introduction

Apart from the significant advances in computer technology and progress in disciplines relevant for solving optimization problems, practical complex problems are still challenging in the sense that it is hard to solve realistically large instances in reasonable computation times. On the top of that issue, there is a question of configuring solver's parameters. The

performance of most meta-heuristic methods is tightly connected with the right choice of their parameters, resulting with analyses that involve yet another optimization problem. One possible approach for dealing with such issues is empirical analysis, in most of the cases connected with the concrete application and implementation.

Bee colony optimization is a population-based meta-heuristic method that was first proposed by Lučić and Teodorović in 2001 [13]. The inspiration for creating new multi-agent system, such as BCO, originates from foraging behavior of the honey bees. This behavior is suitable for modeling since the practice of collecting and processing nectar is highly organized. 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 modification of complete solutions. To provide better understanding of the BCO structure, the introduction into the behavior of the bees in nature is being presented.

In nature, honey bees succeed to find nectar among limited resources in quite efficient manner, without control of some central management and within unpredictable and dynamic environment. The reason for such a success is the capacity for communication using skills that most resemble to symbolic language [10]. It was Karl von Frisch that in the mid-1940s first recognized the *waggle dance* [18], for which he earned Nobel Prize in 1973. The bees are using waggle dance to learn about various properties of food source, such as the position defined by the direction and the distance.

Basically, a mathematical model of the foraging behavior of honey bees can be described as follows. 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, typically in the neighborhood of their hive, are called *scout bees*. 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, 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 will follow one of the dancing scout bees to the previously discovered location. After returning to the hive with a load of nectar, 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 return to its role of an *uncommitted* follower [3, 4]. Although it is yet unknown how an uncommitted bee decides among several recruiters, the fact is that "the loyalty and recruitment among bees are always a function of the quantity and quality of the food source" [8, 17].

2. The BCO Algorithm

The BCO method is based on engagement of a group of *artificial bees* (B individuals) in search for the optimal solution [8]. The homogeneity of artificial bees is being presumed, where each bee generates one solution to the problem. The homogeneity implies that, unlike worker bees in nature, all the artificial bees in BCO are involved in foraging process. 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 criteria is being satisfied. In regard to this clear distribution of tasks for artificial bees, each iteration of the BCO algorithm can be represented as a composition of alternating phases (steps): *forward pass* and *backward pass*.

During the forward pass, all the artificial bees are performing the exploration independent from each other, and therefore, no information is being exchanged in this phase. The method of exploration depends on the implementation of the corresponding BCO algorithm, that is, choice of heuristic. The exploration is performed through certain (predefined) number of moves to either construct the part of a solution [7] or modify the existing complete solution [6]. The number of moves within one forward pass can be represented as a function of the parameter NC. The parameter NC represents the second parameter of the BCO method and its values are influencing the exploitation of the search. Typically, it is used to determine the frequency of information exchange between bees, that is, the number of forward/backward passes during one iteration. At the end of the forward pass the new (partial or complete) solution is generated for each bee [8].

During the backward pass of the BCO algorithm all the artificial bees share the information about the discovered solutions. The information being exchanged in the BCO algorithm contains the quality of each (partial) solution, with respect to the best and the worst solution. Each artificial bee decides, with a probability depending on the solution quality, whether it will stay *loyal* to its solution or not. The artificial bees that stay loyal to their solutions are becoming *recruiters*. Artificial bees that are not loyal to their current solutions, become *uncommitted*, and have to select among the solutions advertised by the recruiters. The se-

lection 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 exploration. Consequently, within each backward pass all bees are being divided into two groups: recruiters, and uncommitted bees.

Initialization: Read input data.
Do
(1) Assign a(n) (empty) solution to each bee.
(2) *For* $(i = 0; i < NC; i++)$
// **forward pass**
(i) Perform move for each bee.
// **backward pass**
(ii) Evaluate the (partial/complete) solutions;
(iii) Loyalty decisions;
(iv) *If* (bee not loyal), choose a recruiter by roulette wheel.
(3) Evaluate all solutions. Update x_{best} and $f(x_{best})$
While stopping criterion is not satisfied.
return $(x_{best}, f(x_{best}))$

Figure 1: Pseudo-code for BCO

The pseudocode of the BCO algorithm is given in Fig. 1. Steps (i) and (ii) are problem dependent and should be resolved in each implementation of the BCO algorithm. On the contrary, other steps of the BCO are problem independent. These steps specify loyalty decision (step (iii)) and recruiting process (step (iv)), and are therefore described in more detail in the following text.

2.1 Loyalty Decision

In order for bees to share information about the quality of discovered (partial) solutions, the BCO algorithm is running through three stages: 1. Evaluation; 2. Loyalty decision; and 3. Recruitment. If value C_b $(b = 1, 2, ..., B)$ denotes the evaluation value of the b-th bee (partial) solution, then it is being normalized to the $[0, 1]$ interval in such a way that larger normalized value O_b corresponds to the better (partial) solution. Usually the evaluation is implemented so that it corresponds to the formulation of the objective function [15].

In the next stage of backward pass the loyalty functions allow, for bees who start exploration from different points in the search space, to decide whether to become uncommitted followers, or to continue exploring already known solutions. The probability that b-th bee (at the beginning

of the new forward pass) is loyal to its previously generated (partial) solution can be expressed as follows:

$$p_b^{0,u+1} = e^{-\frac{1-O_b}{u}}, \quad b = 1, 2, \ldots, B, \tag{1}$$

where parameter u corresponds to the forward pass counter. In this form equation (1) 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). Moreover, as the search process advances the influence of the already discovered (partial) solution increases, i.e., the probability that bee will keep and advertise its current solution has larger value.

Until recently, loyalty function $p_b^{0,u+1}$ was most often used when dealing with optimization problems. From an analytical perspective, it can be reasoned that its utilization agrees well when the search process is implemented so that often interruptions during backward pass should be avoided. In other words, when it is obvious that the search path of a bee will most probably lead to a good solutions, then increasing its loyalty during one iteration assists well such endeavor. However, when the emphasis should be on the exploration of the solution space, different perspective into the measure of bees loyalty needs to be considered. In recent work [16] it was reported that for some variants of BCO, better performance could be achieved if the current forward pass index (u) was omitted in the loyalty decision process. Some other probability functions were examined in [14]. A new study was therefore conducted for 10 different loyalty functions:

(1) $p_b^{0,u+1} = e^{-\frac{1-O_b}{u}}$,
(2) $p_b^1 = e^{-O_{\max}-O_b}$,
(3) $p_b^2 = O_b$
(4) $p_b^{3,n_{it}} = e^{-(1-O_b)/n_{it}}$,
(5) $p_b^{4,u+1} = e^{-(1-O_b)/\sqrt{u}}$,
(6) $p_b^{5,u+1} = e^{-(1-O_b)\sqrt{u}/\sqrt{u+1}}$,
(7) $p_b^{6,u+1} = e^{-(1-O_b)/\log u}$,
(8) $p_b^{7,u+1} = e^{-(1-O_b)/u\log(u+1)}$,
(9) $p_b^8 = e^{-2*(1-O_b)}$,
(10) $p_b^{9,u+1} = e^{-(1-O_b)\log(u+1)/\log(u+2)}$.

Two classes of loyalty functions can be distinguished: *Class I*, as a function of parameter O_b ($p_b^{1,2,8}$), and *Class II* as a two variable function of O_b and counter u or iteration counter n_{it} ($p_b^{0,3,4,5,6,7,9}$).

2.2 Recruiting Process

The probability that b's (partial) solution would be chosen by any uncommitted bee depends on the solution quality of a recruiter b and

equals to:

$$p_b = \frac{O_b}{\sum_{k=1}^{R} O_k}, \quad b = 1, 2, \ldots, R, \tag{2}$$

where O_k represents normalized value for the objective function of the k-th advertised partial solution and R denotes the current number of recruiters.

3. Sensitivity Analysis of BCO

Designing a BCO method in principle includes choosing a procedure for constructive/improvement moves, an evaluation function and setting BCO parameters to a certain values that are usually determined by some set of pilot studies, some previously published work or even intuition. However, the analysis of different settings for loyalty function is lacking in current literature, even though it is a part of the generic section of the BCO method and is not problem specific. One of our goals was to address this issue.

Empirical analysis of the meta-heuristic method belongs to interdisciplinary research and in many cases can require great effort due to the stochastic nature of the method or, in some cases, tuning large number of parameters whose interaction should be expected [11]. Unlike specific orientated optimization algorithms, meta-heuristics methods are categorized by its parameters and/or different modules [2]. Such structure, when implemented, can expose different behavior as parameter values are changing. During the last decade, different tools for experimental analysis were proposed and/or inspected, most of them based on modeling response values with linear or nonlinear models and/or implementing three basic steps: screening, experimentation and exploitation [1, 9, 19]. The literature on this topic today is overwhelming, so the right choice of the tunning method for BCO remains one of the future challenges.

The aim of this work is to provide first insights on behaviour of BCO by following guidelines of many researches who were concerned with methodical empirical analysis of heuristic and meta-heuristic methods. A thorough scientific testing of BCO method can be computationally too extensive, which is why first steps into empirical analysis of BCO is addressing questions of *sensitivity analysis*. Sensitivity analysis corresponds to analysis of variation of algorithm's response values, such as quality of solution (usefulness, utility) or time of execution, while changing its parameter configuration [12]. We examined here a constructive variant of BCO algorithm and used a simple scheduling problem as an example.

Problem formulation. Let m be the total number of identical processors engaged, and n number of non-preemptive independent tasks. The considered scheduling problem consists of assigning tasks to processors, and determining their starting times. Let $T = \{1, 2, \ldots, n\}$ be a given set of independent tasks, where each tasks $i \in T$ has to be processed by exactly one among the identical processors from the set $P = \{1, 2, \ldots, m\}$. Each processor can process only one tasks at the time, and once the tasks has started it will continue to run on the same processor until completion. Let l_i be the processing time of task i $(i = 1, 2, \ldots, n)$, which is known and fixed. The goal is to find a schedule of tasks on processors such that the corresponding completion time of all tasks is minimized. The mathematical programming formulation of the problem can be found in [7], together with the implementation of BCO algorithm that was used in this work. Problem here is referred to as finding minimal *makespan*.

Problem instances. Instances used for testing BCO algorithm represent randomly generated instances with known optimal solutions [7]. They were introduced in [5] for Multiprocessor Scheduling Problems with Communications Delays. The test instances are named as Iogra$< n >_< m >$, where n designates number of tasks, and m denotes number of processors (graph density was set to zero). Specifically, in the work of Davidović et al [7] different problem-size instances were used, i.e., $m = \{2, 4, 6, 8, 9, 10, 12\}$ and number of tasks ranging from 100 to 500. It was concluded that n does not influence the complexity of the problem, as confirmed in new studies. Additionally, new results have shown that the influence of the varying number of processors on the complexity of the problem is not so straightforward. Structure of these problem instances is introduced using box-plots and presented in Fig. 2.

4. Results

Evaluation function $f_b^1 = y_{max}$, introduced in the paper of Davidović et al. [7] depends only on the value of the makespan (y_{max}), and therefore is more receptive due to its lower computational costs. Newly proposed function $f_b^2 = y_{max}/L'$ depends on two parameters, makespan and the current sum of computational time of non-scheduled tasks L'. With introduction of evaluation function f_b^2 better partial solution was associated with larger values (maximality principle), which suggested new course of how the problem can be solved while maintaining objective of minimizing makespan. To best describe new approaches, a concept of *methods of evaluations* was introduced in order to illustrate different evaluation of partial solution in the backward pass of BCO. In case of f_b^1

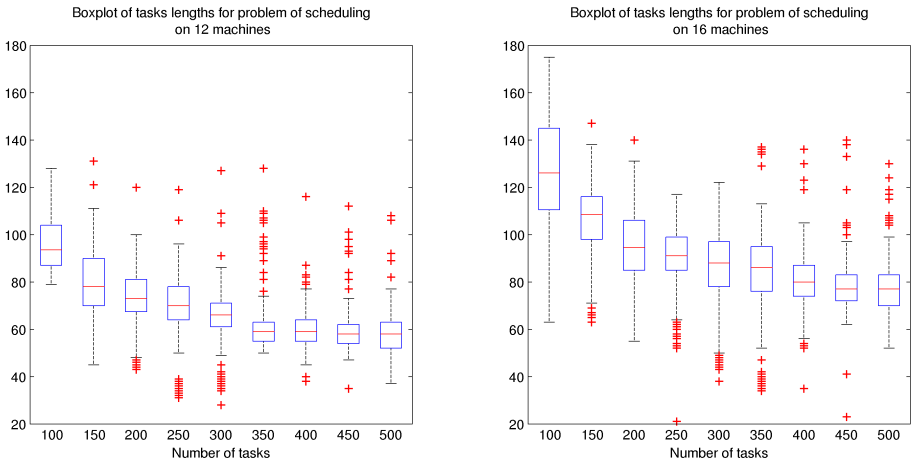

Figure 2: Box-plot for $m = 12, 16$ and 9 instances in relation to the n, where possible outliers are marked with red crosses.

both maximization and minimization principles can be used, thus yielding two different methods of evaluations. When partial solution with biggest makespan is evaluated as the best, it's normalized value is equal to 1, while the solution with lowest makespan will be appointed with normalized value 0. This method of evaluation is denoted here as max, f_b^1. In case of minimization, partial solution with lowest makespan is marked as the best among the population of solutions, and its normalized value corresponds to 1, while maximal makespan will be normalized to value 0. Such method of evaluation is denoted here as min, f_b^1. Justification for incorporating these methods comes from initial set of studies when it was recognized that the solution with smaller makespan doesn't necessarily lead to best result and that both minimal and maximal makespan can be used to quantify good partial solution.

An experiment will be considered as a set of 100 independent runs of the investigated algorithm. All experiments were conducted for predefined values of parameters. In case of the test instances Iogra100_12/16 the experiments were accomplished on complete parameter's search space (Fig. 3). However, for instances where $n \geq 150$ the experiments were conducted on sub-regions of parameters search space. Such restriction comes from limitations imposed on values of NC as they are dependent on number of constructive moves in BCO instance. For example, as the number of tasks increases, maximal number of forward/backward passes also increases which greatly expands the parameters search subspace $S \subseteq \mathcal{B} \times \mathcal{NC}$. Since experimental analyses should be conducted

for each pair (B, NC) it would be too time consuming to include values for $NC > 100$. The choice of values for maximal number of bees was determined arbitrary. Domains of all BCO parameters are provided in Table 1.

Table 1: Parameter space for experimental analysis of BCO.

Parameter	**Domain**
method of evaluation	min, f_b^1; max, f_b^1; max, f_b^2
loyalty function	$p_b^i,\ i \in [0, .., 9]$
B	$[1, 20]$
NC	$[1, 100]$

In each run of an experiment the solution quality was measured within the stopping criteria defined as maximum number of iterations, while maximal number of iterations was set to 100. The set of results used to conduct sensitivity analysis of the BCO parameters is being presented in Fig. 3.

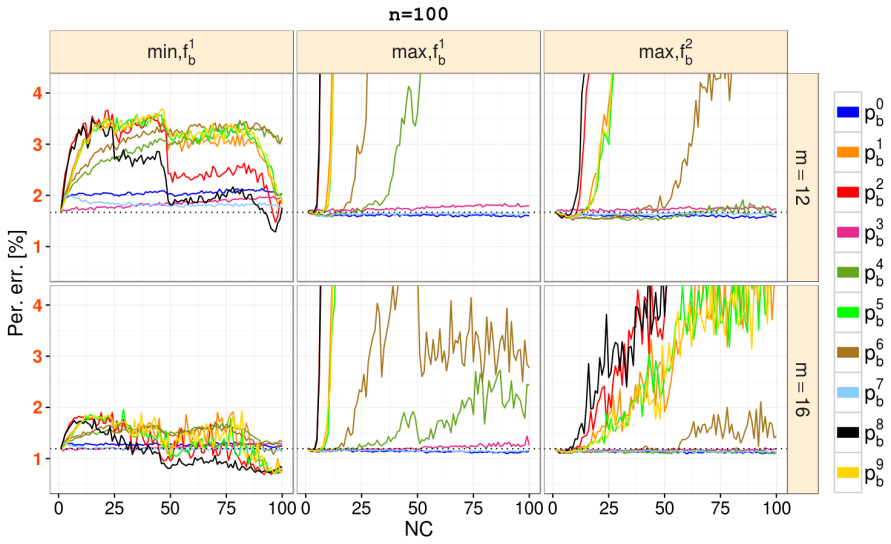

Figure 3: The influence of parameter NC on the averate solution quality in regard to the method of evaluation and loyalty function.

Graphics in Fig. 3 illustrate influence of different BCO parameters on reported average solutions quality, measured by percent error, for two problem instances of different class and same number of tasks. The

main goal of this presentation was to visually inspect improvements in the solution quality when parameter NC changes its value in respect to structural parameters of BCO. As three methods of evaluations were used, graphics are arranged to distinguish their influence on each loyalty functions when $NC \in [1, 100]$. Specifically, each color on a plot corresponds to different loyalty function, whereas dashed black line signifies reference (start) case when $NC = 1, B = 20$. Reference case is used to simulate the behaviour of an underlying heuristic. All values on the graphics correspond to a number of bees that generated best results. It should be noted that parameter B can take different values when reporting on the best value. For example, on problem instance Iogra100_12 and for method of evaluation min, f_b^1, presented profiles of average results are mostly generated when $B = 20$. On the same instance, the best average result in the case of configuration (max, f_b^1, p_b^9) was achieved when $B = 18$. Actually the best solutions were generated when large population of bees was utilized ($B \geq 18$).

There are few interesting observations drawn from Fig. 3. First, for some cases of loyalty functions the influence of methods of evaluation is not distinguishable, as it is the case for $p_b^{0,3,7}$. The reason for such behaviour comes from the fact that these three loyalty functions can converge fast toward cases where the majority of the partial solutions will be transfered to the next iteration. It is most likely that the search is stranded in the local minimum. Therefore, it can be concluded that loyalty functions $p_b^{0,3,7}$ show robustness to changes of other qualitative and quantitative BCO parameters, however, without significant improvement in the solution quality when compared with reference case.

Unlike previous group of loyalty functions, some do not converge fast ($p_b^{4,5,6,9}$) or not converge at all ($p_b^{1,2,8}$). Such a property can yield bigger perturbations in the reported solution quality since the number of partial solutions that will be used for exploitation, varies throughout an iteration. This variation depends on the utilization of evaluation function (or method of evaluation) and the structure of problem instance. Although showing significant fluctuations, those loyalty functions are able to bring improvements into the solution quality, at least for minimization of f_b^1. Among them, loyalty functions p_b^2 and p_b^8 perform the best in respect to the starting case $NC = 1$.

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. However, this is beyond the scope of this paper.

Since the results were sensitive to the choice of problem instance, it was obvious that additional analysis on the whole considered set of problem instances should be undertaken. Such presentation was then used to determine a robust set of parameters configurations that would generate the best results. For this reason a group of graphics presenting the influence of BCO parameters on different problem instances is given in Fig. 4. As before, the value for B varies in interval $[18, 20]$ when generating good quality soutions, with one exception where $B = 15$ was reported by function p_b^8 on problem instance Iogra400_12. The series of graphics on Fig. 4 consists of Fig. 3 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 $n > 100$ due to high computational cost. However, we can still notice similarities between the graphics from different groups, and draw similar conclusions as in case of $n = 100$. First paramount conclusion is related to Class II type of loyalty functions, such as p_b^k, $k \in \{0, 3, 4, 5, 6, 7, 9\}$. From this group, loyalty functions p_b^0, p_b^3 and p_b^7 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. Additionally, they have generated practically insignificant improvements, and as such do not represent good choice for the BCO method on considered set of problem instances. Remaining Class II loyalty functions showed high sensitivity to utilization of method of evaluation and problem instance. No pattern was able to be identified in respect to NC that would generate good solutions. Actually, only p_b^5 and p_b^9 succeeded to be better then the starting $NC = 1$ case but solely on instances Iogra100_12/16, Iogra150_16, Iogra200_12, Iogra250_16 and Iogra300_12/16. Also, these two loyalty functions exhibit similar behavior throughout the search. Loyalty functions of Class I, p_b^1, p_b^2, p_b^8, showed very high sensitivity to changes in quantitative values of B and NC and choice of problem instance. Between these three, the most unsuccessful was p_b^1 and generated solutions that resemble those of $p_b^{5,9}$. Loyalty functions p_b^2 and p_b^8 are the only one that presented certain pattern for values of NC which brings improvements with respect to the starting case $NC = 1$.

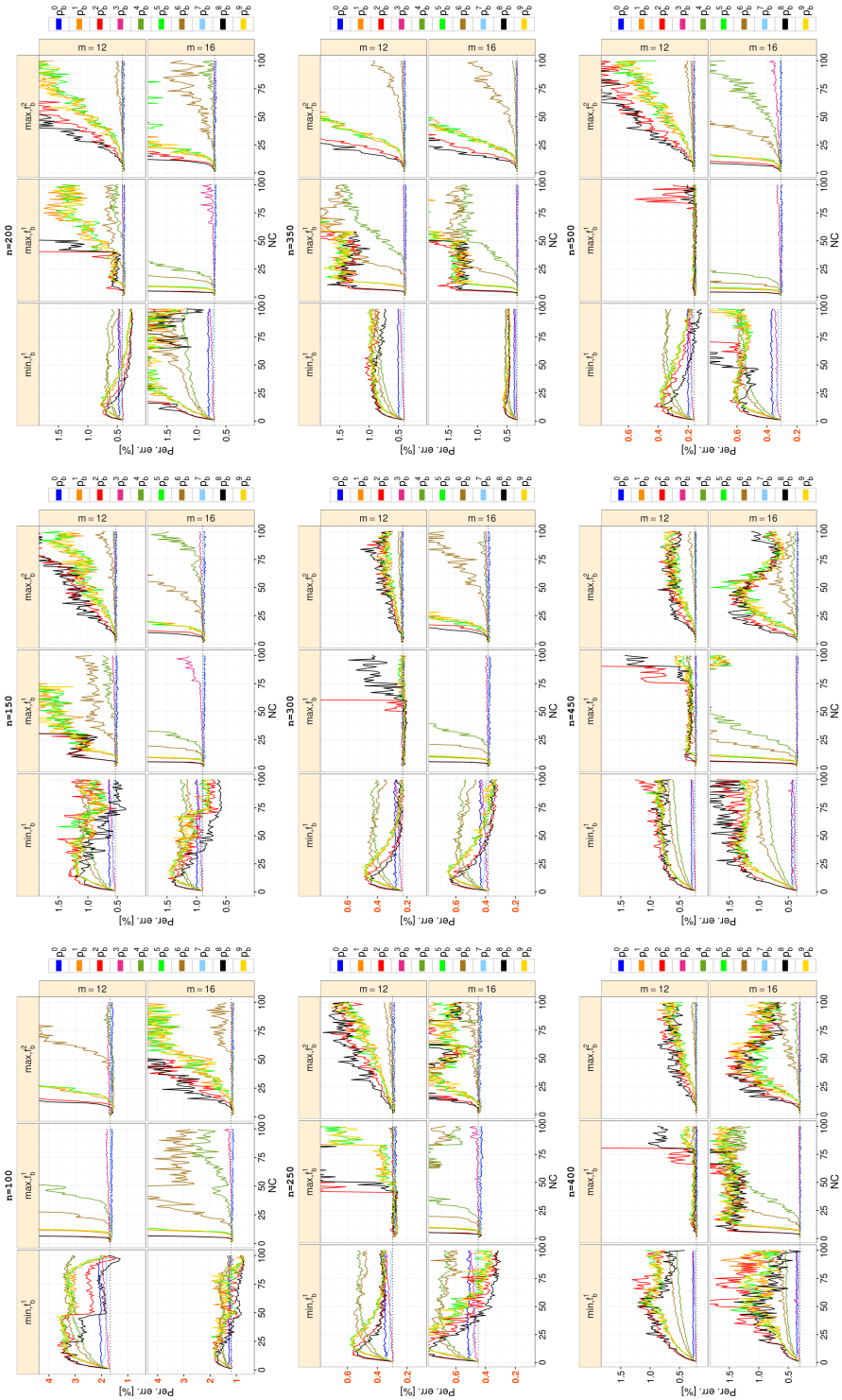

Figure 4: Illustrating the influence of method of evaluation, loyalty function and parameter NC on the performance of BCO on problem of scheduling independent tasks on identical processors.

5. Conclusion

We have tested the influence of different BCO method's parameters on the quality of solutions. In total four BCO parameters were analyzed: B, NC, method of evaluation and loyalty function. Sensitivity analysis was conducted by means of visual inspection of series of graphics categorized by the type of problem instance. Furthermore, each graphic consists of set of plots that reveal the influence of loyalty function with regard to method of evaluation, for fixed values of bees and varying number of parameter NC.

Conducted empirical analysis showed that on provided set of problem instances in 50% of cases best results were obtained for minimization of f_b^1. Furthermore, good quality was obtained for larger population of bees, that is $B \in [18, 20]$, and when $NC \geq 90$. The most successful loyalty functions were those of Class I, p_b^2 and p_b^8 in particular, which were not (often) used previously in the literature. We can conclude that on considered benchmark set of problem instances, configurations $\{min, f_b^1, p_b^{2,8}, B \in [18, 20], NC \geq 90\}$ were most successful, being the only one to offer significant improvements in the solution quality in comparison with reference case. Some additional tests indicate that successful values of NC can be restricted to $[0.9n, n]$, which has yet to be confirmed.

The possible directions for future work could be implementing some of the tuning methods mentioned in [9], on constructive and improvement versions of BCO.

Acknowledgment: This research has been supported by Serbian Ministry of Education, Science and Technological development, Grant. Nos. 174010, 174033, 174008, 044006.

References

[1] T. Bartz-Beielstein and M. Preuß. Experimental analysis of optimization algorithms: Tuning and beyond. In *Theory and Principled Methods for the Design of Metaheuristics*, pages 205–245, Springer, 2014.

[2] M. Birattari. *Tuning metaheuristics: a machine learning perspective.* Studies in Computational Intelligence, vol. 197. Springer, 2009.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems.* Oxford, 1999.

[4] S. Camazine and J. Sneyd. A model of collective nectar source selection by honey bees: self-organization through simple rules. *Journal of Theoretical Biology*, 149(4):547–571, 1991.

[5] T. Davidović and T. G. Crainic. Benchmark-problem instances for static scheduling of task graphs with communication delays on homogeneous multi-

processor systems. *Computers & Operations Research*, 33(8):2155–2177, 2006.

[6] T. Davidović, D. Ramljak, M. Šelmić, and D. Teodorović. Bee colony optimization for the p-center problem. *Computers & Operations Research*, 38(10):1367–1376, 2011.

[7] T. Davidović, M. Šelmić, D. Teodorović, and D. Ramljak. Bee colony optimization for scheduling independent tasks to identical processors. *Journal of Heuristics*, 18(4):549–569, 2012.

[8] T. Davidović, D. Teodorović, and M. Šelmić. Bee Colony Optimization Part I: The Algorithm Overview. *Yugoslav Journal of Operational Research*, 25(1):1367–1376, 2014.

[9] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[10] J. L. Gould. Honey bee recruitment: the dance-language controversy. *Science*, 189(4204):685–693, 1975.

[11] H. H. Hoos and T. Stützle. *Stochastic local search: Foundations & applications*. Elsevier, 2005.

[12] H. H. Hoos and T. Stützle. Empirical analysis of randomized algorithms. In T. F. Gonzalez (Ed.) *Handbook of Approximation Algorithms and Metaheuristics*, Chapter 14, Chapman & Hall/CRC, 2007.

[13] P. Lučić and D. Teodorović. Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence. *Preprints of the TRISTAN IV*, pages 441–445, 2001.

[14] P. Maksimović and T. Davidović. Parameter Calibration in the Bee Colony Optimization Algorithm. *Proceedings of the 11th Balkan Conference on Operational Research*, pages 263–272, 2013.

[15] Z. Michalewicz and David B. Fogel. *How to solve it: modern heuristics*. Springer, 2004.

[16] M. Nikolić and D. Teodorović. Empirical study of the Bee Colony Optimization (BCO) algorithm. *Expert Systems with Applications*, 40(11):4609–4620, 2013.

[17] D. Teodorović. Bee colony optimization (BCO). *Innovations in swarm intelligence*, Studies in Computational Intelligence, vol. 248, pages 39–60. Springer, 2009.

[18] K. von Frisch. Decoding the language of the bee. *Science*, 185(4152):663–668, 1974.

[19] X.-S. Yang, S. Deb, M. Loomes, and M. Karamanoglu. A framework for self-tuning optimization algorithm. *Neural Computing and Applications*, 23(7-8):2051–2057, 2013.