Convergence Analysis of Swarm Intelligence Metaheuristic Methods

Tatjana Davidović(✉) and Tatjana Jakšić Krüger

Mathematical Institute of Serbian Academy of Sciences and Arts, Belgrade, Serbia
{tanjad,tatjana}@mi.sanu.ac.rs

Abstract. Intensive applications and success of metaheuristics in practice have initiated research on their theoretical analysis. Due to the unknown quality of reported solution(s) and the inherently stochastic nature of metaheuristics, the theoretical analysis of their asymptotic convergence towards a global optimum is mainly conducted by means of probability theory. In this paper, we show that principles developed for the theoretical analysis of Bee Colony Optimization metaheuristic hold for swarm intelligence based metaheuristics: they need to implement learning mechanisms in order to properly adapt the probability rule for modification of a candidate solution. We propose selection schemes that a swarm intelligence based metaheuristic needs to incorporate in order to assure the so-called *model convergence*.

Keywords: Optimization problems · Solution quality
Nature-inspired methods · Asymptotic properties · Stochastic processes

1 Introduction

Let us consider an optimization problem that requires minimization of a real-valued function f on a feasible space $\mathcal{X}$. More precisely, $f : S \to \mathbb{R}$ with a domain $S \subseteq \mathbb{R}^n$. S is also called a *set of solutions* (or *solution space*) for the considered optimization problem, while each $\boldsymbol{x} \in S$ represents a *solution*. A solution $\boldsymbol{x} = (x^1, x^2, \ldots, x^n)$ is an array in the n-dimensional space and it consists of components x^i, $i = 1, \ldots, n$. $\mathcal{X} \subseteq S$ is called a *set of feasible solutions* and it contains only the solutions that satisfy constraints defined within the considered optimization problem, i.e., $\boldsymbol{x} \in \mathcal{X}$ represents a *feasible solution*. An *optimal solution* (or *optimum*) of the considered optimization problem is $\boldsymbol{x}^* \in \mathcal{X}$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$. All other solutions $\boldsymbol{x} \in \mathcal{X}$ are called *sub-optimal*. The optimal solution may not exist and then the considered optimization problem is unfeasible. If the optimal solution exists it may not be unique, and therefore, the solving of the considered optimization problem means finding one or more (or sometimes even all) of its optimal solutions.

Finding the optimal solutions is usually a very hard task and it involves the application of exact methods that are time and/or space consuming. The efficient, problem-specific methods designed to find sub-optimal solutions very fast

A. Eremeev et al. (Eds.): OPTA 2018, CCIS 871, pp. 251–266, 2018.
https://doi.org/10.1007/978-3-319-93800-4_20

are called heuristics. Metaheuristic methods have been developed to eliminate limitations of exact and heuristic methods, i.e., to satisfy requirements for less computational resources and to obtain a better quality of sub-optimal solutions at the same time. Today we distinguish various types of metaheuristics [25]. The population-based methods explore the idea that combining or modifying existing solutions can produce new and (hopefully) better ones. Swarm Intelligence (SI), especially its engineering stream, is a discipline of Artificial Intelligence (AI) that studies actions of individuals in various decentralized systems [1]. It explores the behavior of natural entities (consisting of many individuals) in order to build artificial systems for solving problems of practical relevance [4]. Therefore, among population-based we distinguish SI-based metaheuristics such as Ant Colony Optimization (ACO) [6], Artificial Bee Colony (ABC) [18], Bee Colony Optimization (BCO) [3], and Particle Swarm Optimization (PSO) [26]. An exhaustive list of SI-based metaheuristics methods may be found in [21].

Although practically very useful, metaheuristic algorithms suffer from a theoretical disadvantage: it is hard to evaluate the quality of the reported solution. The obtained solution may be even optimal, however, it is almost impossible to prove that. In the literature, metaheuristics are primarily investigated experimentally, usually associated with their concrete engagement and implementations. In addition, some theoretical research related to the convergence analysis of metaheuristic methods has already been conducted [2,10,20,22,28].

The importance of metaheuristics' theoretical background has inspired our work on proving convergence properties of the SI-based metaheuristic methods. The difficulty of theoretical analysis of stochastic search methods, in general, may be found in complex, highly nonlinear and stochastic correlations between their constituting parts [28]. Furthermore, theoretical analysis of metaheuristics commonly implies mathematical verification of the asymptotic convergence of the reported solution towards an optimal one, under some predefined conditions. Assuming that a considered optimization problem is solvable, investigating convergence properties of a metaheuristic algorithm is related to the question: is the optimal solution reachable if the algorithm is given enough time and resources [5]. Inspired by [10,12], two types of convergence for BCO, the so-called *best-so-far* convergence and sophisticated *model* convergence, have been utilized in [15,16]. Therefore, our intention now is to apply the gained insights to all SI-based metaheuristics. The main contributions of this paper are fourfold. First, we systematically review the existing notations and definitions related to the model convergence of metaheuristic methods. Then we provide an extension of the generic procedure (proposed in [10]) in such a way that it reflects the main steps of SI metaheuristic methods. Third, we recommend learning rules that assure model convergence of SI methods. Finally, we provide a systematic proof of model convergence of SI methods towards a global optimum when the recommended learning rules are applied.

In Sect. 2 we review some of the known results related to the SI methods and their convergence analysis. After a short survey of general notation and properties of stochastic sequences in Sect. 3, in Sect. 4 we present conditions

that are sufficient to obtain convergence of SI-based metaheuristics to desired optimal solution. The last section contains concluding remarks.

2 Swarm Intelligence Metaheuristic Methods

From the biological perspective, swarm behavior (such as fish schools, flocks of birds, herds of land animals, insects' communities, etc.) is founded on existential needs of individuals to collaborate without any central control. In such a way they increase the probability to survive because predators mostly attack isolated individuals. This type of behavior is first and foremost characterized by autonomy, distributed functioning and self-organization. With this main idea in mind, SI investigates cooperation of individuals in biological systems and implements them to solve various practical problems [1].

Typical examples of SI metaheuristic algorithms are: ACO, PSO and various bees algorithms. Practicality and usefulness of these algorithms verified experimentally have motivated their theoretical research. Related to PSO, theoretical verification of convergence may be found in [7,17,26,27,29], for ACO in [8,9,12,19,24,30] and for BCO in [15,16]. In the above mentioned papers conditions of convergence are given for the specific method or a specific implementation of the corresponding SI metaheuristic methods. In the case of BCO, the authors of [15,16] have presented sufficient conditions for convergence of a constructive version of the BCO algorithm (BCOc). Theoretical analysis of the improvement-based version of the BCO algorithm (BCOi) is given in [14]. Therefore, our goal is to recognize conditions of asymptotic convergence under a general framework that describes all known SI metaheuristics w.r.t. to both types of generating solutions (constructive and the improvement one).

Constructive SI methods are building solutions by adding components to an empty solution or to already generated partial solutions. The examples of constructive methods are ACO and early versions of BCO, called BCOc. On the contrary, the improvement-based ones are modifying the existing complete solutions in an attempt to improve their quality. Typical examples of improvement-based SI methods are PSO, and BCOi, while ABC represents combination of these two approaches.

2.1 Instance- and Model-Based Algorithms

In [30] the authors proposed a framework that should improve the performance of majority of metaheuristic methods from a theoretical aspect. This framework is based on analyzing parameters of a metaheuristic method. Borrowing the notation from the machine learning field, the authors of [30] recognize two types of metaheuristic methods: *instance-based* and *model-based*. To generate new candidate solutions an instance-based algorithm utilizes only a current solution or a set of current solutions. On the contrary, model-based algorithms utilize a parameterized probabilistic scheme (called model) to generate candidate solutions.

A metaheuristic method is said to satisfy the *model-based search properties* if it iteratively explores the following two steps [30]:

- Generates (constructs or transforms) candidate solutions using some parameterized probabilistic model.
- Modifies the model (i.e., instantiate *update rule*) using candidate solutions such that the search is directed towards more promising regions.

Such a metaheuristic adopts the *model-based parameter scheme*, thus establishing a basis for the *model convergence* [10]. It requires learning properties which may be implemented in a form of an *update rule* for the method's parameters and/or structure [30,31]. The update rule represents the utilization of information extracted during the search in order to update the model.

2.2 Generic Procedure

The authors of [10,11,30] have presented a general framework called *generic algorithm* (i.e., *generic procedure*) to encompass most (or all) known metaheuristics for combinatorial optimization problems. The generic procedure allows a rather flexible description of different segments (modules) of a metaheuristic. It can be viewed as an iterative algorithm that utilizes two different structures: (1) m_t - a state of memory, and (2) L_t - a list of N *sample points*, i.e., solutions ($\boldsymbol{x}_s \in \mathcal{X}$, $1 \leq s \leq N, N \in \mathbb{N}$) in iteration t. The stopping criterion is defined as the maximum number of iterations. The main steps of the generic procedure are as follows:

1. $t \leftarrow 1$;
2. Initialization of memory m_t;
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 $f(\boldsymbol{x}_s)$ for all $\boldsymbol{x}_s \in L_t$ and generate a list L_t^+ of pairs $(\boldsymbol{x}_s, f(\boldsymbol{x}_s))$;
   (c) Determine new memory state m_{t+1} as a function $h(m_t, L_t^+, \xi_t')$ of current memory state m_t, current list L_t^+ and random influence ξ_t';
   (d) $t \leftarrow t + 1$.

The state of memory m_t may be further defined by two components: the so-called *sample-generating part* (m_t^s) and the *reporting part* (m_t^r). The sample-generating part holds all necessary information to generate L_t in iteration t. All other relevant information are stored in m_t^r, such as solution of the highest quality found so far, namely the *best-so-far* solution, $\boldsymbol{x}^{bsf}$. The procedure to determine $\boldsymbol{x}^{bsf}$ depends on the best solution found in iteration t, i.e., on the *iteration best* solution ($\hat{\boldsymbol{x}}_t$). The solution $\hat{\boldsymbol{x}}_t$ is determined as $f(\hat{\boldsymbol{x}}_t) = \min_{1 \leq s \leq N} f(\boldsymbol{x}_s)$. We should emphasize that for the purpose of convergence analysis $\boldsymbol{x}^{bsf}$ represents a current approximation of an optimal solution [10]. Function $g(m_t, \xi_t)$ determines a probability distribution of new solutions to enter the list L_t, while function

$h(m_t, L_t^+, \xi_t')$ defines rules to determine the state of memory m_{t+1} in the next iteration. Moreover, $g(m_t, \xi_t)$ refers to a set of possible transformations (i.e., possible moves) that generate new solutions depending on the current model. Function $h(m_t, L_t^+, \xi_t')$ is responsible for modifying (updating) the model by integrating learning properties ([10], p. 168). It is important to note that in this formalism, information about a problem instance may be used as an argument of functions g and h. Consequently, the update rules differentiate w.r.t. the type of optimization problem being solved and the type of heuristic rules that either construct or modify solutions during the search. The provided generic procedure accommodates well all notable SI metaheuristic algorithms as it was shown for some versions of ACO, PSO [10] and BCO [14].

Here, we present a modification of generic procedure that contains more details about the steps of typical SI method. Finding a solution of an optimization problem requires all of its components to be determined. Constructive methods are selecting values for components and building a solution step by step, while the improvement-based metaheuristics are transforming the current values of solution components in order to improve the quality of the considered solution. The SI methods that satisfy model-based search properties involve some learning steps that influence the solution generation process. More precisely, the determination of solution components values is influenced by the quality of previously generated solutions. The main focus of learning is the modification of selection scheme for values of components (expressed by selection probability values $p_{i,j}$). As we are considering the combinatorial optimization problems, the set of possible components values is finite (or at most countable). Consequently, selection probability $p_{i,j}$ measures the chances that component i will take value j. If the SI method is model-based the values for selection probabilities will change from iteration to iteration, i.e., we will have different values for $p_{i,j}(t),\ t = 1, 2, \ldots$

SI metaheuristics are population-based methods, and therefore their m_t consists of N current solutions, best-so-far solution, current selection probability values, and some other data specific for each particular method (we do not go into details here and consider only the first three items). The pseudo-code of model-based SI metaheuristic method is presented by the Algorithm 1.

Each SI metaheuristic method has its specificities, however, they can be described by some general steps. At the beginning, the reading of problem data and setting of parameter values is performed. Then, some initialization is required, and we described it in the following way: best-so-far solution (the final solution to be reported to the user at the end of execution) is initialized to an empty solution and the corresponding objective function value is set to a large enough constant. These steps are not necessary yet they ensure the compactness of our pseudo-code. The values for selection probabilities $p_{i,j}(1)$ are set to given initial values, denoted by $p_{i,j}(0)$, $\nu(i)$ denotes the number of possible values for the component i. The values $p_{i,j}(0)$ may be different in various SI methods, although usually all selection probabilities have the same initial value. Some of the methods involve restarts in their executions and then the values of $p_{i,j}(t)$ are set to $p_{i,j}(0)$ again. We also assume that the initial state of

```
t ← 1
Read(Problem input data, method parameters)
x^bsf ← ∅, f(x^bsf) ← ∞ {initialization of m_t}
for i ← 1, n do
   for j ← 1, ν(i) do
      p_{i,j}(t) ← p_{i,j}(0)
   end for
end for
{main loop}
repeat
   for s ← 1, N do
      x_s ← CreateSolution(m_t, p_{i,j}(t), rand)
      if f(x_s) < f(x^bsf) then
         x^bsf ← x_s
      end if
   end for
   for i ← 1, n do
      for j ← 1, ν(i) do
         p_{i,j}(t + 1) ← Update(p_{i,j}(t))
      end for
   end for
   t ← t + 1
until stopping criterion is satisfied
Return(x^bsf, f(x^bsf))
```

Algorithm 1: Pseudo-code for SI method

m_t (actually the content of m_1) does not include any solution from the population. This enables to capture the characteristics of both constructive and improvement-based methods. As our REPEAT loop is executed for $t = 1$ first, each particular method can generate initial population according to its own set of rules (realized by procedure CreateSolution). The same (or similar) procedure can be used in all remaining iterations to generate the new population of solutions by the considered SI method and possibly update $\boldsymbol{x}^{bsf}$. The final step of each iteration consists of updating values for selection probabilities to be used in the next iteration.

3 Convergence Analysis

Convergence of a stochastic sequence addresses the question whether or not a series of random variables $(X_1, X_2, \ldots)$ converges to a new random variable X^*. In case of metaheuristic algorithms we observe a sequence of solutions produced at the end of each iteration. Having in mind that SI metaheuristics are population-based methods, we need to determine a single solution $\boldsymbol{x}_t = (x_t^1, x_t^2, \ldots, x_t^n) \in \mathcal{X}$ that is reported at the end of iteration t. Here, $x_t^i \in \mathbb{R}$ represents the i-th component of the solution $\boldsymbol{x}_t$. Usually, $\boldsymbol{x}_t = \hat{\boldsymbol{x}}_t$.

Because the algorithm incorporates a global knowledge exchange among the iterations, we are able to obtain best-so-far solution $\boldsymbol{x}_t^{bsf}$ in any iteration t. Here, we use a strict policy for the update of $\boldsymbol{x}_t^{bsf}$. Namely, in the initialization phase of the search the value of variable $\boldsymbol{x}^{bsf}$ is set arbitrary until the condition $f(\hat{\boldsymbol{x}}_t) < f(\boldsymbol{x}^{bsf})$ is satisfied and then $\boldsymbol{x}^{bsf}$ copies a value from $\hat{\boldsymbol{x}}_t$. Consequently, we become interested in the sequence of solutions $\boldsymbol{x}_t^{bsf}$ generated by the metaheuristic method, i.e., we observe the sequence $(\boldsymbol{x}_1^{bsf}, \boldsymbol{x}_2^{bsf}, \ldots, \boldsymbol{x}_t^{bsf}, \ldots)$.

The best-so-far convergence analyzes conditions under which the sequence $(\boldsymbol{x}_t^{bsf})_{t=1}^{\infty}$ converges to an optimal solution $\boldsymbol{x}^*$, more precisely, it evaluates probability that an optimal solution will be found at least once during the search. Accordingly, one should find the conditions that guarantee a sequence of objective function values $f(\boldsymbol{x}_t^{bsf})$ to converge ("w. pr. 1" or "in probability") to $f^* = \min\{f(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}\}$. As shown in [10,23,24] the only requirement is that, in any iteration, (p^*) – the probability to find an optimal solution is strictly greater than zero. However, the concept of best-so-far convergence is too "generous" as it appears that even a random search, known as quite inefficient algorithm, converges to a global optimum [23]. A superior behavior may be expected only in the cases when the set of current solutions (also referred to as model) tends to be modified into the set of optimal and high quality solutions, as $t \to \infty$. This type of modification is named the *model convergence* [10] and it tends to evaluate the probability that the algorithm reaches a state in which it generates only optimal solution(s). Model convergence is hard to prove as it requires adequate balance between exploration and exploitation of the search obtained by fine-tuning of the algorithm's parameters.

4 Model Convergence of SI Optimization Methods

In this section we present conditions that are sufficient for any SI based metaheuristic algorithm to find an optimal solution. We start with necessary notation and remind on known theoretical results in the literature. The basic tools of probability theory are utilized, such as limit theorems for stochastic sequences, in particular the second Borel-Cantelli lemma which we do not repeat here.

To express the model-based search properties, the SI method needs to be well organized (structured) algorithm which utilizes the information about the performance from the previous stages of its execution. Consequently, the global knowledge exchange between iterations is the main assumption in our analysis. These requirements are fulfilled if CreateSolution procedure of the SI generic pseudo-code (Algorithm 1) includes some learning properties.

4.1 Preliminary Conditions

To explain the course of our analysis, the following events should be defined (borrowing the notation from [10,13]).

Definition 1. *For any feasible solution* $\boldsymbol{x}$ *of the considered problem and an iteration counter* $t \geq 1$ *assume that:*

- $C(t)$ *denotes the event that* $\boldsymbol{x}_t = \boldsymbol{x}$*, i.e., the examined solution was generated in iteration* t*.* $C^c(t)$ *is used to represent the complementary event.*
- $B(t)$ *marks the event that* $\boldsymbol{x}$ *was not visited during the first* t *iterations i.e.,* $B(t) = C^c(1) \cap C^c(2) \cap \cdots \cap C^c(t)$*. The complementary event is* $B^c(t)$*.*
- $B = \bigcap_{t=1}^{\infty} B(t)$ *represents the event that the algorithm cannot generate the examined solution* $\boldsymbol{x}$*, i.e.,* $\boldsymbol{x} \neq \boldsymbol{x}_t$ *for all* $t = 1, 2, \ldots$
- $r(t) = \Pr(B^c(t)|B(t-1)) = \Pr(C(t)|B(t-1))$ *means the probability that* $\boldsymbol{x}$ *is generated in the iteration* t*, although it has not been obtained in any of the previous iterations.* ◊

According to Definition 1, $B(1)$ denotes that $\boldsymbol{x}$ was not produced by the algorithm in the first iteration, $B(2)$ describes the situation that $\boldsymbol{x}$ was not visited in the first and second iteration, and so on. Consequently, $\{B(t)\}_{t=1}^{\infty}$ represents a non-increasing sequence of events[1], more precisely:

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

Based on Definition 1 and definition of convergence in probability from [10, 14,16], $\Pr(C(t)) \to 1$ as $t \to \infty$ denotes that the sequence $\boldsymbol{x}_t$ converges in probability to the set $\mathcal{X}^*$ (that contains only optimal solutions). Moreover, based on the definition of events B and $B(t)$ we can conclude that

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

To establish connection between events introduced in Definition 1 we present here the theorem about the convergence of Generalized Hill Climbing algorithm (GHC) proven in [13]. The pseudo-code for GHC is presented by the Algorithm 2 in Appendix.

Theorem 1. *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* $t \to \infty$.

Then, an equivalent form for (i) can be shown, i.e.,

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

Lemma 1 was obtained for the single-solution metaheuristic, however, it is straight-forward to extend it to the population-based methods by observing the sequence of best-so-far solutions.

[1] In [10] the author mistakenly reported that the sequence $\{B(t)\}_{t=1}^{\infty}$ is non-decreasing.

In order to identify the model convergence properties for majority of the SI metaheuristic methods, we need to define four modification schemes for the values of selection probabilities. Based on our previous experience, besides constructive and improvement-based SI metaheuristics we need to distinguish between two types of optimization problems. The first type includes optimization problems for which the size of the solution is smaller then the size of the problem. More precisely, these are the so called selection problems (like p-median or p-center location problems). In these kind of problems we are usually given a number of possibilities to choose a subset of their values that will constitute a solution of the considered problem. In this case, modification scheme for selection probability values needs to consider only components' affiliation within $\boldsymbol{x}^{bsf}$. The second type of optimization problems are the ones whose solutions represent orderings and/or groupings of all given elements. For these problems the length of the solution vector is equal to the size of the problem. Typical examples of the second type problems are Traveling Salesman Problem (TSP), Vehicle Routing Problem (VRP) and various scheduling problems. For these problems the probability update rules should be modified in such a way to take care of ordering or group affiliation of solution components.

4.2 Modification Schemes in the Cases When Subset of Data Constitutes a Solution

For the constructive SI method the selection probability for component i taking value j in the iteration $t+1$ should be modified in the following way:

$$p_{i,j}(t+1) = \begin{cases} 1-\lambda_t \cdot (1-p_{i,j}(t)) & \text{if } j \in \boldsymbol{x}^{bsf}; \\ \lambda_t \cdot p_{i,j}(t) & \text{if } j \notin \boldsymbol{x}^{bsf}; \\ p_{i,j}(0) & \text{if } j \text{ was not chosen before.} \end{cases} \tag{1}$$

where $0 < \lambda_t \leq 1$ represents the time dependent *learning rate*. As we already mentioned, $p_{i,j}(0)$ represents the initial value for selection probability. The idea is to learn the influence of the component's value to the quality of generated solutions. This means that if value j belongs to the best-so-far solution, the probability that j will be included as a value for component i of some solution constructed in the next iteration increases. If the value j is not in the best-so-far solution but was selected before (in some of the previous iterations) the probability of its selection in the next iteration decreases. For all values that were not considered in any of previous iterations the selection probability value remains unchanged (its value is still equal to the initial one).

In addition, we define the probability of generating an optimal solution for this type of problems by the constructive SI method as the following indicator function of the pair (i, j):

$$p^*_{i,j} = \begin{cases} 1 \text{ if } j \in \boldsymbol{x}^*; \\ 0 \text{ otherwise.} \end{cases}$$

In the case of the improvement-based metaheuristic method the selection probability that component i should take value j in the iteration $t+1$ should be modified as follows (assuming $0 < \lambda_t \leq 1$):

$$p_{i,j}(t+1) = \begin{cases} 1-\lambda_t \cdot (1-p_{i,j}(t)) & \text{if } j \in \boldsymbol{x}^{bsf} \text{ and } i \notin \boldsymbol{x}^{bsf}; \\ \lambda_t \cdot p_{i,j}(t) & \text{otherwise;} \end{cases} \tag{2}$$

where $i \notin \boldsymbol{x}^{bsf}$ is an abbreviation for $x_s^i \notin \boldsymbol{x}^{bsf}$ and it actually refers to the previous value of the component i in any solution $\boldsymbol{x}_s$ considered for transformation. If value j was a part of the best-so-far solution and the current value of component i was not, the probability that j will substitute the current value of component i in the next iteration is increased. In all other cases we decrease the probability of selecting value j in the next iteration.

The corresponding indicator function is defined in the following way:

$$p_{i,j}^* = \begin{cases} 1 \text{ if } i \notin \boldsymbol{x}^*, j \in \boldsymbol{x}^*; \\ 0 \text{ otherwise.} \end{cases}$$

4.3 Modification Schemes in the Cases When All Data Constitute a Solution

First, we consider the problems (such as TSP) requiring to properly order solution's components. The selection probability modification schemes in the case of constructive and improvement-based SI methods have the same form:

$$p_{i,j}(t+1) = \begin{cases} 1-\lambda_t \cdot (1-p_{i,j}(t)) & \text{if } (i,j) \in \boldsymbol{x}^{bsf}; \\ \lambda_t \cdot p_{i,j}(t) & \text{otherwise;} \end{cases} \tag{3}$$

where $0 < \lambda_t \leq 1$. Here, we calculate the probability that component i should be assigned value j either by adding value in a constructive method or by transforming its previous value in an improvement-based method. The notation $(i,j) \in \boldsymbol{x}^{bsf}$ describes the case that components $i-1$ and i are getting the same combination of values as in the best-so-far solution. The corresponding indicator function can be defined as follows:

$$p_{(i,j)}^* = \begin{cases} 1 \text{ if } (i,j) \in \boldsymbol{x}^*; \\ 0 \text{ otherwise.} \end{cases}$$

The same rule is applied to the VRP type problems, while for scheduling like problems pair (i,j) denotes that the task i should be allocated to the group j. Consequently, the modification (selection) scheme (3) is applicable in this case as well.

4.4 Sufficient Conditions for Model Convergence of SI Methods

Let us assume that the considered SI algorithm is applying one of previously defined schemes for modifying selection probabilities. In this section we provide

the sufficient conditions for convergence in probability of the SI obtained solution toward an optimal one. In order to assure model convergence of the SI algorithm, two conditions should be satisfied: (i) all feasible solutions need to be reachable from any initial solution and (ii) upon an optimal solution is found its generation has to be favored.

The first condition represents the best-so-far convergence and it is satisfied if, with probability one, there exists an iteration $t > t_0$ in which any considered solution $\boldsymbol{x}$ (optimal included) is found. This is consistent with the condition (i) from the Theorem 1, i.e., the probability that an optimal solution will never be generated tends to zero when $t \to \infty$. The second condition is related to model convergence. It requires to prove that, after generating an optimal solution, by applying the corresponding update rule defined by one of the Eqs. (1), (2) or (3), the generation of optimal solutions will be supported for the considered SI algorithm. This actually means that $p_{i,j}(t)$ converges to $p^*_{i,j}$.

Theorem 2. *The conditions*

$$1 \geq \lambda_t \geq \frac{\log t}{\log(t+1)} \text{ for all } t \geq t_0, \ (t_0 \geq 2), \tag{4}$$

and

$$\sum_{t=1}^{+\infty}(1-\lambda_t) = +\infty, \tag{5}$$

are sufficient for the corresponding SI method to converge in probability toward an optimal solution $\boldsymbol{x}^*$ *from* $\mathcal{X}^*$.

Proof: (i) (best-so-far convergence) We actually prove that $\Pr(B) = 0$, i.e., the equivalent condition from Lemma 1. Let $\boldsymbol{x}$ be a given feasible solution. Then $C(t)$ means that $\boldsymbol{x}$ is found for the first time in iteration t. As

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

then it holds

$$\begin{aligned}\Pr(B) &= \Pr(\{C^c(1) \cap C^c(2) \cap \cdots\}) \leq \Pr(\{\boldsymbol{x} \text{ is never found}\}) \\ &= \prod_{t=1}^{+\infty} \Pr(\{\boldsymbol{x} \text{ is not found in iteration } t | \boldsymbol{x} \text{ is not found in iteration } k < t\}).\end{aligned} \tag{6}$$

If we refer to solution components and selection probability update rules (1), (2) and (3), in the worst case for all pairs of components (i, j) not being established in iterations $1, \ldots, t$, it holds:

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

(justifiable easily by induction). Applying the first condition of the Theorem 2, it holds

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

In such a way, for any pair of components (i,j), we obtained a lower bound of the worst case selection scenario. Consecutively, even in the worst case, for the probability to find the solution $\boldsymbol{x}$ by the considered SI method it holds:

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

n being the number of components in the solution $\boldsymbol{x}$.
Assuming that the solution $\boldsymbol{x}$ was not found by the SI method, an upper bound on the right hand side of the relation (6) is obtained:

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

Applying logarithm and the convexity of the exponential function indicated by $(1-a)\leq e^{-a}, \forall a\in(0,1)$ (i.e., $\log(1-a)\leq -a$), from the previous term we obtain:

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

Now we deduce that

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

i.e., $\Pr(B)\leq 0$ in (6). As $\Pr(B)\geq 0$ always holds, it is obvious that $\Pr(B)=0$.

(*ii*) (Model convergence) Let us assume that $\boldsymbol{x}^*$ is generated in the iteration m for the first time. Then $\boldsymbol{x}_t^{bsf}=\boldsymbol{x}^*$ for all $t\geq m$. Moreover, we can prove that in all iterations $t>m$ the selection probability for pairs (i,j), not included in $\boldsymbol{x}^*$, decreases, i.e., converges to zero as $t\to\infty$.

Let $(i,j)\notin\boldsymbol{x}^*$. According to (1), (2) and (3), the selection probability of pair (i,j) in iteration $m+r$, $r=1,2,...$ will be modified as follows:

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

The previous claim can be verified easily by induction. In addition, the condition (5) gives us

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

Consequently, for the probability that $(i,j) \notin \boldsymbol{x}^*$ will be used again after the optimal solution $\boldsymbol{x}^*$ was generated, it holds:

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

That is exactly what needed to be proved for model convergence part of the theorem. ■

5 Conclusion and Future Work

We provide the sufficient conditions for the model convergence of the SI-based metaheuristics: (1) all feasible solutions must be reachable from any point in the solution space; (2) once an optimal solution is found, its generation is favored. To fulfill these requirements, the SI algorithm needs to incorporate selection schemes that exploit the knowledge from the previous search. We have provided three modification schemes w.r.t. the characteristics of the SI method and the considered optimization problem. Although the established conditions guarantee the asymptotic convergence of the SI generated solution toward one of the optimal ones, the question of practical usability of this result still remains open. Namely, in practice we cannot perform an infinite number of iterations, and therefore, we need also the evaluation of convergence speed. Consequently, future work should include runtime analysis of an expected time (iteration index) to obtain the optimal solution, the so-called *first hitting time*.

Acknowledgments. This research was supported by the Serbian Ministry of Education, Science and Technological Development, Grant Nos. 174033, 044006 and F159.

Appendix

To describe the GHC algorithm, we remind on several definitions as provided in [13]. An objective function $f : S \to [0, +\infty)$ is defined on a finite set S of all possible solutions. Two important components of GHC are: (a) *neighborhood function* $\eta : S \to 2^S$, where $\eta(\boldsymbol{x}) \subseteq S$, for all $\boldsymbol{x} \in S$, and (b) *hill climbing random variables* $R_k : S \times S \to \mathbb{R}$, where $k = 1, 2, ...$ indicates an iteration counter of the outer loop controlled by STOP_OUTER. The pseudo-code for GHC algorithm is given as Algorithm 2. At each iteration i of GHC's inner loop, a candidate solution $\boldsymbol{x}$ is generated uniformly at random among all neighbours of the solution $\boldsymbol{x}_i \in S$ i.e., according to probability mass (density) function $h_{\boldsymbol{x}_i}(\boldsymbol{x}) = 1/|\eta(\boldsymbol{x})|$.

The hill climbing random variables R_k are utilized to accept or reject neighbour solution. STOP_INNER is the stopping criterion for inner loop utilized to inspect if a current solution is a local optimum and if the counter k should be incremented. According to [13], it is important to assume that the verification if the current solution is a local optimum can be conducted in polynomial time. This is possible if there are a polynomial number of neighboring solutions of the current solution. As a consequence, the generation of a local optimum implies that a new random variable will be used within the next iteration.

```
Define function η and a set of random variables R_k;
k ← 1; /* initialize outer loop counter */
i ← 0; /* initialize inner loop counter */
Select an initial solution x_0 ∈ S;
while not STOP_OUTER do
  while not STOP_INNER do
    Generate x ∈ η(x_i) according to h_{x_i}(x);
    Δ(x_i, x) = f(x) − f(x_i);
    if R_k ≥ Δ(x_i, x) then
      Accept solution x;
    else
      Reject solution x;
    end if
    i ← i + 1;
  end while
  k ← k + 1;
  Update R_k;
end while
```

Algorithm 2: Pseudo-code of the GHC algorithm.

It is important to note that the search space S must be *reachable*, that is, all solutions are accessible regardless of a starting point $\boldsymbol{x}_0$ [13].

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
2. Chen, J., Ni, J., Hua, M.: Convergence analysis of a class of computational intelligence approaches. Math. Probl. Eng. **2013**, 1–10 (2013)
3. Davidović, T., Teodorović, D., Šelmić, M.: Bee Colony Optimization- Part I: the algorithm overview. Yugoslav J. Oper. Res. **25**(1), 33–56 (2015)
4. Dorigo, M., Birattari, M.: Swarm intelligence. Scholarpedia **2**(9), 1462 (2007). http://www.scholarpedia.org/Swarm_intelligence
5. Dorigo, M., Blum, C.: Ant Colony Optimization theory: a survey. Theor. Comput. Sci. **344**, 243–278 (2005)

6. Dorigo, M., Stützle, T.: Ant Colony Optimization: overview and recent advances. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics. ISOR, vol. 146, pp. 227–263. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-1665-5_8
7. Garcia-Gonzalo, E., Fernandez-Martinez, J.L.: A brief historical review of Particle Swarm Optimization (PSO). J. Bioinform. Intell. Contr. **1**(1), 3–16 (2012)
8. Gutjahr, W.J.: A graph-based ant system and its convergence. Future Gener. Comput. Syst. **16**(8), 873–888 (2000)
9. Gutjahr, W.J.: ACO algorithms with guaranteed convergence to the optimal solution. Inf. Process. Lett. **82**(3), 145–153 (2002)
10. Gutjahr, W.J.: Convergence analysis of metaheuristics. In: Maniezzo, V., Stützle, T., Voß, S. (eds.) Matheuristics: Hybridizing Metaheuristics and Mathematical Programming. AOIS, vol. 10, pp. 159–187. Springer, Boston (2009). https://doi.org/10.1007/978-1-4419-1306-7_6
11. Gutjahr, W.J.: Stochastic search in metaheuristics. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics. ISOR, vol. 146, pp. 573–597. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-1665-5_19
12. Gutjahr, W.J.: Ant colony optimization: recent developments in theoretical analysis. In: Auger, A., Doerr, B. (eds.) Theory of Randomized Search Heuristics: Foundations and Recent Developments, pp. 225–254. World Scientific (2011). https://www.worldscientific.com/worldscibooks/10.1142/7438
13. Jacobson, S.H., Yücesan, E.: Global optimization performance measures for generalized hill climbing algorithms. J. Global Optim. **29**(2), 173–190 (2004)
14. Jakšic Krüger, T.: Development, implementation and theoretical analysis of the Bee Colony Optimization metaheuristic method. Ph.D. thesis, University of Novi Sad (2017)
15. Jakšić Krüger, T., Davidović, T.: 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)
16. Jakšić Krüger, T., Davidović, T., Teodorović, D., Šelmić, M.: The Bee Colony Optimization algorithm and its convergence. Int. J. Bio-Inspired Comput. **8**(5), 340–354 (2016)
17. Jiang, M., Luo, Y., Yang, S.: Stochastic convergence analysis and parameter selection of the standard Particle Swarm Optimization algorithm. Inf. Process. Lett. **102**(1), 8–16 (2007)
18. Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N.: A comprehensive survey: Artificial Bee Colony (ABC) algorithm and applications. Artif. Intell. Rev. **42**, 21–57 (2014)
19. Kötzing, T., Neumann, F., Röglin, H., Witt, C.: Theoretical analysis of two ACO approaches for the Traveling Salesman Problem. Swarm Intell. **6**(1), 1–21 (2012)
20. Liu, H., Abraham, A., Snásel, V.: Convergence analysis of swarm algorithm. In: NaBIC, pp. 1714–1719 (2009)
21. Parpinelli, R.S., Lopes, H.S.: New inspirations in swarm intelligence: a survey. Int. J. Bio-Inspired Comput. **3**(1), 1–16 (2011)
22. Pintér, J.: Convergence properties of stochastic optimization procedures. Optimization **15**(3), 405–427 (1984)
23. Solis, F.J., Wets, R.J.B.: Minimization by random search techniques. Math. Oper. Res. **6**(1), 19–30 (1981)
24. Stützle, T., Dorigo, M.: A short convergence proof for a class of Ant Colony Optimization algorithms. IEEE Trans. Evol. Comput. **6**(4), 358–365 (2002)

25. Talbi, E.G.: Metaheuristics: From Design to Implementation. Wiley, New York (2009)
26. Trelea, I.C.: The Particle Swarm Optimization algorithm: convergence analysis and parameter selection. Inf. Process. Lett. **85**(6), 317–325 (2003)
27. Van Den Bergh, F.: An analysis of Particle Swarm Optimizers. Ph.D. thesis, University of Pretoria (2006)
28. Yang, X.-S.: Metaheuristic optimization: algorithm analysis and open problems. In: Pardalos, P.M., Rebennack, S. (eds.) SEA 2011. LNCS, vol. 6630, pp. 21–32. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20662-7_2
29. Zeng, J.C., Cui, Z.H.: A guaranteed global convergence Particle Swarm Optimizer. J. Comput. Res. Dev. **8**, 1333–1338 (2004)
30. Zlochin, M., Birattari, M., Meuleau, N., Dorigo, M.: Model-based search for combinatorial optimization: a critical survey. Ann. Oper. Res. **131**(1–4), 373–395 (2004)
31. Zlochin, M., Dorigo, M.: Model-based search for combinatorial optimization: a comparative study. In: Guervós, J.J.M., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., Fernández-Villacañas, J.-L. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 651–661. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45712-7_63