Hindawi
Journal of Mathematics
Volume 2018, Article ID 8072621, 14 pages
https://doi.org/10.1155/2018/8072621

*Research Article*

# The Influence of Problem Specific Neighborhood Structures in Metaheuristics Performance

**A. S. Santos,[1] A. M. Madureira,[1] and M. L. R. Varela [2]**

[1]*Interdisciplinary Studies Research Center (ISRC), School of Engineering, Polytechnic Institute of Porto (ISEP/IPP), Portugal*
[2]*Department of Production and Systems, University of Minho, Portugal*

Correspondence should be addressed to M. L. R. Varela; leonilde@dps.uminho.pt

Received 26 February 2018; Revised 12 May 2018; Accepted 7 June 2018; Published 4 July 2018

Academic Editor: Biswajit Sarkar



Metaheuristics (MH) aptitude to move past local optimums makes them an attractive technique to approach complex computational problems, such as the Travelling Salesman Problem (TSP), but there is lack of information on the parameterization procedure and the appropriate parameters to improve MHs' performance. In this paper the parameterization procedure of Simulated Annealing (SA) and Discrete Artificial Bee Colony (DABC) is addressed, with a focus on the Neighborhood Structure (NS). Numerous NS have been proposed for specific problems, which seem to indicate that the NS is a special parameter, whose optimization is independent of other parameters. The performance of eight NS was examined with SA and DABC under two optimization constraints, regarding computational time variation, to determine if there is one appropriate NS for the TSP problem, independent of the rest of the parameters of the optimization procedure. The computational study carried out for comparing the evaluation of the NS, including a statistical analysis, demonstrated a nonproportional increase in the performance of DABC with some NS. For SA the improvement of the solutions appeared to be more uniform with an almost nonexistent variance in improvement.

## 1. Introduction

In Complex Combinatorial Problems (COPs), the number of constraints and the overall dimension of the problem can make the optimization procedure arduous through enumerative techniques [1, 2]. In order to approach these complex optimization problems researchers have concentrated on decision support systems, web-based systems, or platforms, which integrate several optimization techniques and methods, such as heuristics and metaheuristics. Some examples of optimization techniques are presented in [3–9], which include enumerative heuristic optimization techniques to approach scheduling problems.

Metaheuristics are the predominant techniques to approach COPs, such as OSP (Operational Scheduling Problems), VRPs (Vehicle Routing Problems), or WMSs (Warehouse Management Systems) and their parameterization has a notable impact on their performance.

Software tools developed to solve COPs either use predetermined parameters, not optimized for all instances of the problem, or require the user to determine the parameters themselves. A study on the influence of the parameters can lead to the development of self-parameterization software tools that choose the appropriate parameters for all instances of a problem without the users input. One parameter common to most metaheuristics is the ND (Neighborhood Structure), which is examined in this paper to understand if there is one NS that performs better in all instances of the same problem or not. In other words, should the NS be "parameterized" or can one NS be used for all instances of a problem.

One traditional COP is the Travelling Salesman Problem (TSP), which has extensive applications from transportation to logistics, networking, and commerce [3–9]. Moreover, the TSP has also application in production planning and control (PPC), for instance, for sequencing a set of jobs, to be processed in a single machine environment, under cyclic production. In this PPC problem context the cities correspond to the jobs, and the objective consists on reducing the total production time or makespan, and one well-known algorithm for solving it is proposed by Little et al. [10]. In this paper Simulated Annealing (SA) and Artificial Bee Colony

(DABC) metaheuristics are applied to solve the TSP. However, this paper is less focused on the overall performance of metaheuristics than the impact of the parameterization procedure with special emphasis on the Neighborhood Structure (NS).

Therefore, in this paper the proposed metaheuristics for being analyzed, SA and DABC are applied to the Euclidian-TSP, and the results are presented through a computational study. However, the overall performance of each metaheuristic is not the main purpose of the work. Instead, this paper focused on the NS, which have been characterized as a unique parameter whose optimization is independent of other parameters and the limitation of the optimization procedure, to demonstrate that it should be treated as another parameter of the optimization techniques.

This paper is divided into five sections, which are structured as follows: Section 2 presents the literature review about the classic Travelling Salesman Problem (TSP), the Neighborhood Search Techniques (NST), the metaheuristics (MH) and more specifically the two MHs used in this computational study, which were the Simulated Annealing (SA) and the Discrete Artificial Bee Colony algorithms (DABC), and also the parameterization procedure for the MHs. Additionally, this section finalizes with the description of the eight neighborhood structures (NS) used on the SA and the DABC, whose performance is the focus of this document. Section 3 further describes the problem focused and presents the model underlying the proposed approach. Section 4 presents the computational study carried out in this paper, including the details about the parameterization procedure for both MHs used. Moreover, in this section the problem instances analyzed are detailed, along with the result obtained through the SA and DABC through the use of the eight NS. Lastly this section presents the computational results and a statistical analysis about them, for showing how each NS influences the behavior of both MH considered. Finally, Section 5 summarizes the contribution of this work, by presenting the main conclusions reached along with planned further work.

## 2. Literature Review

*2.1. Travelling Salesman Problem.* One of the best-known optimization problems is the Travelling Salesman Problem (TSP), which can be described as the problem of finding the tour between N cities (or jobs), which means that the tour needs to start and end in the same point and visit each point of the tour once, with the smallest possible distance (or makespan) [3]. TSP is a thoroughly studied problem in the domain of COPs, both as a benchmark problem for optimization techniques as well as the wide amount of applications of the problem, as can be realized through the works presented in [3–9], including the context of production planning and control, and more specifically for jobs sequencing in single machine problems with cyclic production to minimize the makespan, and a well-known Branch-and-Bound algorithm for this purpose was proposed by Little et al. [10]. TSP is a classic NP-hard problem, whose optimization complexity is beyond the plainness of the problem definition. There is no polynomial-time algorithm able to solve TSP, as such there is still much interest on this problem, continuing to motivate researchers to develop efficient algorithms for solving it.

Even though there are several optimal algorithms for the TSP none are polynomial, which leads to MH. Like with most COPs, TSP's MHs are the most used techniques, namely, Simulated Annealing (SA) [3, 9], Tabu Search (TS) [11], Genetic Algorithms (GA) [4], Neural Networks (NN) [5], Ant Colony Optimization (ACO) [6], Honey Bees [12], Particle Swarm Optimization (PSO) [8], and Memetic Algorithms (MA) [13], among others.

For the TSP, the 2-opt and 3-opt have been the most efficient k-opt algorithms [4, 7]. Some researchers recommended that k-opt algorithms should satisfy that $k \leq 3$, to limit the computational time. For TSP, 2-opt has been shown to produce efficient solutions, while 3-opt produced better solutions but is computationally more expensive than 2-opt [14].

One of the best-known formulations of the TSP is the sequential formulation proposed in [15]. In this formulation, $N = \{1, 2, \ldots, n\}$ is the set of cities, $x_{ij}$ are the binary decision variables, where $x_{ij}=1$ if $j$ follows $i$ in the tour; otherwise $x_{ij}=0$, and $c_{ij}$ is the distance between $i$ and $j$. Danzig et al. [15] can be examined as follows; other formulations are presented in [16].

Objective function:

$$Min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

subjected to

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in N \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j \in N \tag{3}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} \leq |M| - 1, \quad \forall M \subset N \ st \ \{1\} \notin M, \ |M| \geq 2 \tag{4}$$

Equation (1) is the objective function that minimizes the salesman tour. Equations (2) and (3) define that each destination is entered once (2) and left once (3). Equation (4) prevents the formation *subtours*, or formation of multiple tours, each with a subset of the $n$ vertices.

One variation of the classical TSP is the Euclidian-TSP. A TSP is called a Euclidian-TSP if the cities or destinations are coordinates in a map and the distance between the cities $i$ and $j$ ($c_{ij}$) is the straight-line distance between $i$ and $j$. It is also important to make the distinction between the Symmetric TSP and the Asymmetric TSP. In the Symmetric TSP for all destinations $i$ and $j$, $c_{ij}=c_{ji}$, while in the Asymmetric TSP at least for one $i$ and $j$, $c_{ij} \neq c_{ji}$ [17].

TSP in the classical sense (Asymmetric TSP) and other variations of the problem, such as the Euclidian Symmetric TSP, have been connected and applied to several practical problems. In [18] the applications of TSP are presented in

$s = s_0$
**While** not Termination Criterion **Do**
  Generate $N(s)$
  **If** $s'$ is not Better than $s$ **Then** Stop
  $s = s'$
**EndWhile**
**Output:** $s$

Algorithm 1: Neighbourhood Search (adapted from [22]).

detail. In the classical sense the TSP is linked to a distribution problem, such as the VRPs (Vehicle Routing Problems), but it can also be used in WMSs (Warehouse Management Systems). The TSP can also be linked to several other COPs, such as the Operational Scheduling Problems (OSPs) [18].

*2.2. Neighborhood Search Techniques.* One of the oldest and best-known sets of optimization methods is based on Neighborhood Search Techniques (NST), which search for the optimal or suboptimal solution of an optimization problem by iteratively manipulating the solutions of the problem as can be realized in [1, 2, 4, 7, 19–21]. In the most straightforward algorithm (the steepest descent algorithm), the solutions will be moved in the direction of the best solution within the current neighborhood. Once there are no solutions within the neighborhood that are better than the current solution the NST procedure will stop. Another NST variant, the "first descent algorithm", moves to the first solution it finds within the neighborhood that is better than the current solution.

Talbi [22] presents a template for a Neighborhood Search Techniques (NST). In Talbi template $S$ represents the set of feasible solutions, $s_0$ the initial solution, $s$ the current solution, and $N(s)$ the neighbors of $s$ and $s'$ the best solution in $N(s)$. An adaption of the Talbi template is presented in Algorithm 1, but there are several other templates for NST.

NSTs search for the optimal solution of COPs with fast and simple to implement procedures, but are often blocked in suboptimal solution (local optimum) due to the limitations of the technique as stated in [1, 2, 4, 7]. In instances where NST cannot overcome suboptimal solutions, heuristics based on techniques developed in Artificial Intelligence domain (AI) could be applied, as can be observed in [1, 2, 4, 7, 23–25]. Unlike NST techniques, metaheuristics have a stochastic component to overcome suboptimal solutions and are able to provide near-optimal solutions without excessive processing time as stated in [1, 2, 4, 7, 25].

*2.3. Metaheuristics.* Metaheuristics are the most used techniques to solve complex computational problems, such as Complex Combinatorial Problems [23]. Metaheuristics are an iterative procedure that controls a subordinate heuristic in the exploration of the totality of the solution space [24, 25]. In order to avoid suboptimal solutions, metaheuristics incorporate disruptive mechanisms that allow them to overcome local optimum. For example, Simulated Annealing allows worst solutions to be accepted as the current solution in an attempt to find the optimum [1, 24]. Genetic Algorithms, like SA, perform stochastic search [1, 2, 23–25]. Simulated Annealing [1, 19] is based on the annealing process in metallurgy, while Genetic Algorithms [1, 2, 23, 26] are based on Darwin's evolutionary theory.

Metaheuristics can be divided into single solution metaheuristics and population metaheuristics [22]. Therefore, the metaheuristics are divided in accordance with the number of solutions on each iteration for the exploration of the solution space. Single solution metaheuristics are inspired in Neighborhood Search Techniques, and population based metaheuristics are driven from the Genetic Algorithms. Generally the single solution metaheuristics have higher intensity than the population based ones, which simultaneously use a number of solutions to explore a larger part of the solution space [24, 25].

Single solution metaheuristics are based on the Neighborhood Search Techniques (NST) with a stochastic component to overcome the local optimum. From the initial solutions, single solution metaheuristics will move the search for better solutions within a neighborhood, as in Neighborhood Search Techniques, but with a stochastic component, which allow them to depart from the predetermined search path and move outside the neighborhood local optimum. Literature identifies several single solution metaheuristics, such as the following [24, 27]:

(i) Greedy Adaptive Search Procedure (GRASP).
(ii) Simulated Annealing (SA).
(iii) Tabu Search (TS).
(iv) Variable Neighborhood Search (VNS).

Population based metaheuristics, unlike single solution metaheuristics, manipulate several solutions, on each iteration, through the optimization procedure. From the initial population, these techniques will manipulate the population to optimize a given performance criterion or a set of criteria. Population based metaheuristics are conventionally more diverse than single solution metaheuristics since these techniques perform a broader search of the solution space. In the literature it is possible to identify several population based metaheuristics such as [24, 27]

(i) Ant Colony Optimization (ACO);
(ii) Artificial Bee Colony (ABC);
(iii) Particle Swarm Optimization (PSO);
(iv) Genetic Algorithms (GA).

Recent research has focused on the development of tools that can better assist enterprises in their convoluted COPs. Advancements in AI have allowed the development of distributed optimization support systems that learn and self-adapt to their environment. In [2, 4, 7, 13] are presented some works based on the applications of AI for distributed optimization systems and in [19] a set of fuzzy optimization problems is solved through Simulated Annealing (SA). Moreover, for more background on uncertain sets please refer to [28–30].

AI-based approaches to address complex COPs use specific knowledge about the problem in the decision procedure as can be realized in the following works [1, 4, 7]. For instance, rule-based approaches store dispatching rules into a knowledge base to assist in the decision procedure, which enables modern enterprises to solve the complex decision problems, from the TSP up to other optimization problems, such as a wide range of manufacturing scheduling problems [4, 7, 24, 25]. AI-based techniques have included metaheuristics to address COPs can be realized, for instance, through the following contributions [1, 2, 24]. Neighborhood Search Techniques (NST) are also useful to approach COPs, since they are easy to implement and use, while enabling effective solutions without excessive computational time, especially in simple COPs as stated in [4, 7, 24, 25].

*2.3.1. Simulated Annealing.* Simulated Annealing is one of the best-known metaheuristics, developed in the 1980s by Kirkpatrick et al. [31] and Cerny [32] inspired in the thermic treatment of metals. Moreover, it is one of the best-known metaheuristics to approach COPs. It is computational efficient as it selects the best possible solution between a finite number of possible solutions, much like Neighborhood Search Techniques (NST) [2]. It adapts the basic NST procedure with a mechanism that allows it to overcome local-optima. Unlike NST, worst solutions are accepted with a predetermined probability dictated by the temperature parameter that controls the likelihood of solutions that are worse than the current one to be accepted as the new current solution, which decreases with the evolution of the search procedure. Since SA can find near-optimal solutions, which in a real environment are often acceptable, it is a particularly attractive technique to approach the TSP as referred to in [3, 9].

Talbi [22] presents a template for Simulated Annealing, where $S$ represents the set of all feasible solutions, $s_0$ the initial solution, $s$ the current solution, and $s'$ a neighbor of the current solution. $T$ represents the current temperature, $T_{max}$ the initial temperature, and $g(T)$ the Cooling Scheme, which is the temperature update. $\Delta E$ is $f(s') - f(s)$, $L$ is the Epoch Length or the number of iteration before the temperature decreases, and $Pi(\Delta E,T)$ is the probability of acceptance of $s'$. An adaption of the Talbi template is presented in Algorithm 2 and for more details on $g(T)$ and $Pi(\Delta E,T)$ please refer to [22]. In this paper, SA will be implemented with a Geometric Cooling Scheme, such as $g(T)=\alpha T$, where $\alpha$ represents the Cooling Ratio.

In this paper the impact of the Neighborhood Structure (NS) in the performance of SA is represented in Algorithm 2 in the underline line. SA was selected because it is one of the best-known metaheuristics and does not require a comprehensive parameterization process.

*2.3.2. Discreet Artificial Bee Colony.* Artificial Bee Colony (ABC) is one of the newest population based metaheuristics. Developed by Karaboga, in 2005 [20] and Pham et al., in 2005 [21], it was inspired in the communal behavior of bees in search for food sources.

**Input:** Cooling Scheme
$s = s_0$
$T = T_{max}$
**Repeat**
  **Repeat**
    **Generate** $s'$
    Calculate $\Delta E$
    **If** $\Delta E \leq 0$ **Then** $s = s'$
    **Else** $s = s'$ with a Probability $Pi(\Delta E,T)$
  **Until** $L$
  $T = g(T)$
**Until** Termination Criterion
**Output:** Best Solution Found

Algorithm 2: Simulated Annealing (adapted from [22]).

Bees' social behavior inspired several metaheuristics, such as the Queen Bee Evolution (QBE), based on colony structure, Marriage in Honey Bees Optimization (MBO) in the reproduction process, and Bee Colony Optimization (BCO) or Virtual Bee Algorithm (VBA), in search of food [27]. ABC was developed for continuous optimization problems, but there are several versions for discrete problems, including the Discrete Artificial Bee Colony (DABC), proposed by Karaboga and Gorkemli to solve the Euclidian-TSP [33]. Since it has been proposed, the DABC has been used to solve the TSP [34], the Operational Scheduling Problem (OSP) [35], and the Manufacturing Cell Design Problem (MCDP) [36]. In [37] the performance of DABC was compared to SA in an OSP. Both metaheuristics were fine-tuned with Taguchi Experiments and in the computational study DABC outperformed SA.

DABC has a similar procedure to ABC. In DABC bees explore discreet solutions within the predetermined Neighborhood Structure. Worker bees explore food sources, opportunistic bees choose and explore food sources, and scout bees search for new food sources [33].

In detail, DABC and ABC repeat three phases, before the interruption criterion, the phase of the worker bees, the phase of opportunistic bees, and the phase of scout bees. First, each worker bee is allocated to each of the initial food sources ($s_0$). In the worker bee's phase, each bee will explore a solution within the neighborhood of their current food source ($s_i$). If the new solution is better it replaces the current found source. Second, opportunistic bees wait on the beehive for the performance of each food source ($s_i$) before each of the opportunist bees chooses a food source. Once the food sources are selected, each bee will explore the neighborhood of the food source. If the new solution is better than the food source, the new solution replaces it. Third, it is the phase of scout bees, if a food source was abandoned, which occurs after $l$ iterations without improvement in the performance of that food source.

$S$ represents all feasible solutions, $s_0$ the initial solutions or initial food sources, $s_i$ the current solution or food sources, and $s_i'$ a neighbor of the solution $s_i$. $P_i$ represents the likelihood of an opportunistic bee moving the food source $s_i$.

**Input:** Cooling Scheme
$s_i = s_0$
**Repeat**
  **For Each** Worker Bee: **Generate** $s_i'$
  **For Each** $s_i$ Calculate $P_i$
  Distribute the Opportunict Bees though $s_i$ with $P_i$
  **For Each** Opportunistic Bee: **Generate** $s_i'$
  **If** $s_i' \leq s_i$ Then $s_i = s_i'$
  **Else** Increase $l_i$
  **If** $l_i = l$
    Tranform the Worker Bee in $s_i$ into Scout Bee
    Scout Bee: Generate $s_i$
    Transform Scout Bee into Worker Bee in $s_i$
**Until** Termination Criterion
**Output:** Best solution found

Algorithm 3: Discrete Artificial Bee Colony.

$l$ is the Limit and establishes the number of iterations without improvement before a food source is discarded and $l_i$ is the number of iterations without improvement in the food source $s_i$. The template for Discrete Artificial Bee Colony is presented in Algorithm 3 and for more detail on $P_i$ please refer to [22].

In this paper the impact of the Neighborhood Structure in the performance of DABC will be analyzed, which is represented in Algorithm 3 in the underline lines. Unlike SA, which does not require a comprehensive parameterization, DABC is more sensitive to the parameterization.

*2.3.3. Parameterization.* Metaheuristics' appropriate parameterization can improve its search technique performance in specific problems. However, the parameterization is often so time consuming that it becomes harder than the implementation of the metaheuristics itself [38]. Hunter et al. [39] present a formal definition of the parameterization, as the process that picks the combination of parameters that optimize the performance of the technique in the problem, which in the case of metaheuristics is particularly important in the determination of how intense/diverse the search is in order to ensure that the optimization procedure is both effective and efficient.

In this paper the study presented in [40] was expanded with the inclusion of four more NS, which enable the insertion of higher diversification into the search procedure. It focuses on the evolution of the performance of the metaheuristics with the increase of the computational time through the eight NS examined, instead of the overall performance of each metaheuristic. Both metaheuristics will be tested with stopping condition of 1 and 10 seconds, to analyze the improvement through each NS, with an increase of the available execution time. Since the NS defines how the search is conducted, it is expected that the relation between the NS and the other parameters will determine how the metaheuristics perform.

Unlike most of the other metaheuristics SA does not require an exhaustive parameterization. Given sufficient time SA will perform well without precise parameters as stated in [41, 42]. However, since the available time is limited, the parameters need to be accurately determined to maximize its performance. SA parameters include the Initial Temperature, the Cooling Ratio, the Epoch Length, the Neighborhood Structure, and the Stoppage Condition.

Kirkpatrick et al. [31] state that at the Initial Temperature (Ti) all new solutions should be accepted. Other authors in [20] state that the Initial Temperature (Ti) values should be between [0.7, 0.8]. Several metrics to determine the Initial Temperature are presented, for instance, in [43, 44]. Park and Kim in [41] examined the impact of each parameter in the performance of SA. According to [41] the Initial Temperature (Ti) should be initiated with a value that accepts almost all movements, which means that the initial acceptance probability (Pi) should be near 1 at the start of the search procedure. Rose et al. in [44] indicate that the Cooling Ratio should slowly decrease the temperature. For a Geometric Cooling Ratio $\alpha$ should be fixed between [0.80, 0.99]. One method to calculate $\alpha$ is presented in [41], which determines the Cooling Ratio based on the Initial and Final Temperatures. It is important to mention that if the Final Temperature is too low the SA will intensify its search too early and turn into a local search procedure. Finally, the Epoch Length (L), which is the number of iterations that should be conducted before the temperature decreases, should be related to the size of the problem instance as referred to in [44]. For example, in an instance of the TSP problem with 50 cities the Epoch Length should be 50.

DABC and subsequently ABC have less parameters than SA [45], but while SA does not need a comprehensive parameterization process, DABC and ABC require a thorough parameterization process to boost their performance. Other than the Neighborhood Structure (NS) and the Stoppage Condition, the other DABC parameters are the Colony Size and the Limit. Kockanat and Karagoba [46] reviewed the impact of the Colony Size ($L$) on the performance of the metaheuristic and determined that ABC is not excessively sensitive on the Colony Size. Akay and Karaboga [47] stated that the Colony Size ($L$) does not need to be outsized especially because this parameter will increase the computational time required before the metaheuristic converges. In [45] it is stated that the Colony Size ($L$) should be linked with the size of the problem, but also it is established that Colony Size ($L$) does not require an exhaustive tuning since its impact on the performance of the metaheuristic is limited. The Limit ($l$) establishes the number of iterations that should occur without improvement of the food source before it is discarded. Several authors [45–48] stated that the Limit ($l$) does not require to be meticulously set for larger $L$. Larger colonies can have smaller limits, since the lack of diversity from smaller limits are balanced by the number of individuals in the population. However, for smaller $L$ the Limit ($l$) needs to be precisely determined to optimize the metaheuristics' performance. In [45] the authors presented some rules to determine the limit, based on the Colony Size ($L$) and the dimension of the problem. There are other parameters considered in some implementations of ABC and DABC, some authors have used a fixed number of scout bees to increase the diversity of the

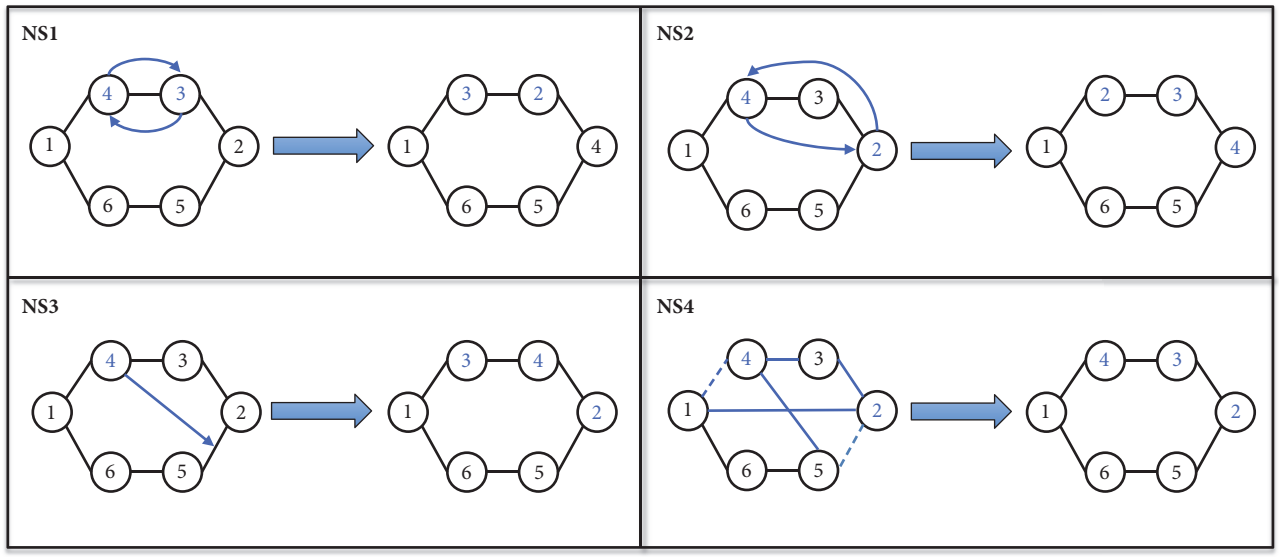


Figure 1: NS1, NS2, NS3, and NS4.

search of the metaheuristic, which was not considered in this implementation of DABC.

*2.4. Neighborhood Structures.* Each metaheuristic has its own parameters, which can influence the balance of how intense and diverse the search is; however, one parameter common to all metaheuristics is the Neighborhood Structure. NS determines how the metaheuristic will move through the solutions space, since it will determine how the current solution is transformed into another solution in one basic movement. In SA, the NS determines how each solution is transformed into another solution, which is then analyzed to determine if it should replace the current solution and in DABC, the NS determines how worker and out looker bees search the area around their current food source. In NST the NS determines the solutions that should be examined on each iteration. Mechanisms that manipulate several solutions, as the crossover mechanism in GA, can be understood as part of a NS.

While NS can be complex, developed to address one specific problem, there are also several simple NS, which broad-scoped mechanics allow them to approach various similar problems. In our proposed model and underlying computational study, the four NS revised in [40] were expanded to eight, which is thought to be able to demonstrate the impact of the NS in the performance of metaheuristics better than [40].

NS1 to NS4 are four common permutation based NS while NS5 to NS8 combine the previous NS. NS1 or *Transpose* is the smallest NS, which consists of all the solutions that can be obtained by interchanging two adjacent elements of the solution [49]. For example, the sequence of {1,2,3,4,5,6}, where the $2^{nd}$ and $3^{th}$ elements were selected, would become {1,3,2,4,5,6}. NS2 or *Swap* is a notorious NS for permutation problems, which consists of all the solutions that can be obtained by interchanging two elements of the solution [49, 50]. For example, the sequence {1,2,3,4,5,6}, where the $2^{nd}$ and $4^{th}$ elements were selected, would become {1,4,3,2,5,6}. NS3 or *Insert*, which consists of all the solutions that can be obtained by removing one element of the solutions and inserting it into another position [49, 50]. For example, the sequence of {1,2,3,4,5,6} where the $2^{nd}$ element is selected and inserted in $4^{th}$ position would become {1,3,4,2,5,6}. NS4 or *2-Opt*, which combines *Swap* and *Insert*, selects two paths, disconnects, and reconnects them in another manner [50, 51]. For example, {1,2,3,4,5,6}, where the paths {1,2} and {4,5} were selected, would become {1,4,3,2,5,6}. In Figure 1 the NS1, NS2, NS3, and NS4 can be analyzed with the examples.

NS5, NS6, NS7, and NS8 combine NS4 and NS1, NS2, NS3, and NS4 into a more ample and diverse NS. NS5 uses NS4 combined with NS1 or *2-Opt* and *Transpose*. NS6 uses NS4 combined with NS2 or *2-Opt* and *Swap*, NS7 uses NS4 combined with NS3 or *2-Opt* and *Insert*, and NS8 does NS4 or two *2-Opt* movements. NS5 to NS8 produce larger neighborhoods, which should result in a more diverse search than NS1, NS2, NS3, and NS4 for both metaheuristics. In Figure 2 the NS5, NS6, NS7, and NS8 can be examined.

## 3. Problem Description and Proposed Model

In this paper the authors want to study how the performance of SA and DABC evolve with an appropriate selection of NS structures and through an increase in the MHs' execution duration. It is important to clarify that the main aim of this paper is not to analyze the overall performance of the metaheuristics itself but, instead, to analyze if the performance improvement of the metaheuristics' is intensified when more diversity is introduced through some kind of proposed NS analyzed and also how this improvement varies when the MHs (SA and DABC) processing time is increased, from 1 to 10 seconds, as it is expected that either the insertions of more diversity through the selected NS and/or through the increase

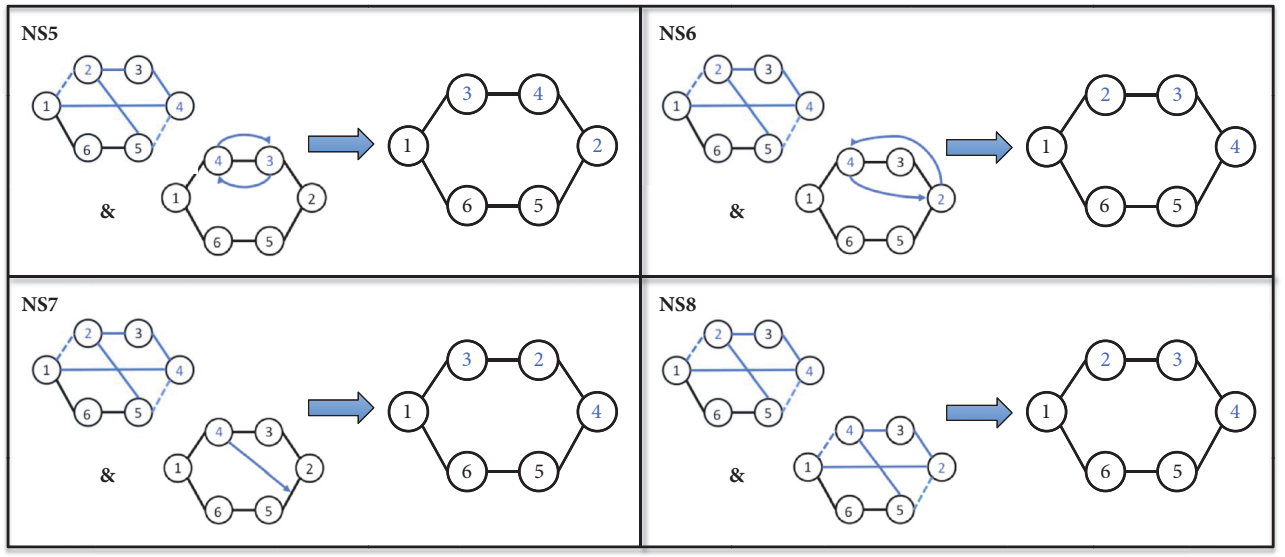


Figure 2: NS5, NS6, NS7, and NS8.

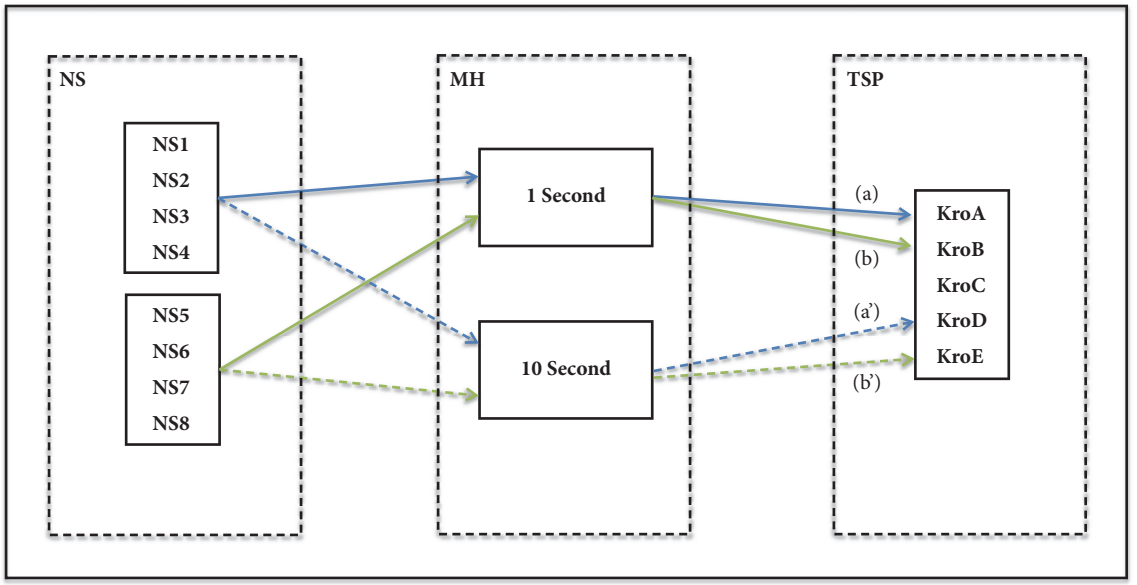


Figure 3: Proposed model.

in the duration of the MHs' execution time will result in more significant improvements in the performance of the MHs for the solution of the TSP.

Therefore, in this paper the impact of the eight NSs considered in the performance of DABC and SA will be studied to determine if there is one appropriate NS for all instances of the TSP problem. Figure 3 represents the model used in the computational study carried out.

In the computational study, the eight NS will be divided into NS1-NS4 that should result in more intense searches and NS5-NS8 that should result in more diverse searches. SA and DABC will be tested with NS1-NS4 and NS5-NS8 with 1 and 10 seconds as the stoppage criterion, which were selected to ensure that a metaheuristics converged to the optimum, in five instances of the Euclidian-TSP available on the TSPLIB [52]. All the results will be normalized into the mean percent deviation from the optimum, which is the best-known metric to benchmark metaheuristics [53], before the median ($\eta$) improvement is examined.

In Figure 3 the median ($\eta$) improvement of NS1-NS4 is $a$-$a'$, and median ($\eta$) improvement of NS5-NS8 is $b$-$b'$. The median ($\eta$) improvement of NS1-NS4 and NS5-NS8 will be tested for SA and DABC with the Wilcoxon Mann-Whitney $U$ test, with the hypothesis:

$$H_0: \eta_{NS1/4} = \eta_{NS5/8}.$$

$$H_1: \eta_{NS1/4} \neq \eta_{NS5/8}.$$

Table 1: Parameters of SA and DABC.

| | SA | | | DABC | |
|---|---|---|---|---|---|
| | Initial Temperature ($T_i$) | Cooling Ratio ($\alpha$) | Epoch Length ($L$) | Colony Size ($L$) | Limit ($l$) |
| **1s** | 2000 | 0.95 | 500 | 20 | 1000 |
| **10s** | 20000 | 0.95 | 500 | 20 | 10000 |

## 4. Computational Study

Both metaheuristics and the eight NS were implemented in C language through Microsoft Visual Studio 2012. The computational tests were performed on a MacBook Pro with a 3GHz Intel® Core i7 processor, 16GB of 1600MHz RAM, and Windows 10 64-bit.

SA and DABC were tested with all the NS with 1 and 10 seconds as the stoppage criterion, in 5 academic benchmark instances of the Symmetric Euclidian-TSP, KroA100, KroB100, KroC100, KroD100, and KroE100, with 100 nodes each, which correspond to (100)! = 9.33262... E157 possible solutions. All the problems are benchmark problems available in the TSPLIB [52], the well-known database of TSPs, which also supplies the optimal solutions for some problems. KroA100, KroB100, KroC100, KroD100, and KroE100 are five instances of the Symmetric 2D-Euclidian-TSP with known optimums, which are 21282 for KroA100, 22141 for KroB100, 20749 for KroC100, 21294 for KroD100, and 22068 for KroE100. Results from each metaheuristic with all the implemented NS were normalized around the known optimal solution and compared to validate if there is statistical evidence of the variations in the performance with the increase of the computational time, which was used as the stoppage criterion for both metaheuristics.

The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

*4.1. Parametrization Used for SA and DABC.* The Stoppage Condition for both MHs, SA and DABC, will be the duration of the optimization procedure, since the purpose of this paper is to evaluate how the performance of the metaheuristics will evolve with each NS when the duration of the search procedure is increased from 1 to 10 seconds. It is expected that when the duration increases, the NS that introduces more diversity will result in a larger improvement.

Table 1 presents the remaining parameters for both 1 and 10 seconds, which cannot be equal or the search would converge too early in the 10 seconds trial. Once more, the focus of this paper is not the overall performance of the MHs, but the evaluation of how the performance of each NS evolves with the increase of the computational time for each MH considered.

SA and DABC parameters were determined for the 1-second runs with the metric referred earlier and adapted for 10-second runs with an increase of the Initial Temperature ($T_i$) in SA and the Limit ($l$) in DABC. All the other parameters were used for both 1 and 10 seconds.

*4.2. Computational Results.* SA and DABC were run five times for each instance of the problem, the best solution was kept while the others were discarded. Both metaheuristics found solutions close to the optimum with 1 and 10 seconds. Figure 4 presents the computational results, the top row represents the result of SA, and the bottom row represents the results of DABC, for 1 and 10 seconds with each Neighborhood Structure represented in a different color.

In SA with 1 second, NS4 found the best solutions in three out of five instances (21459, 22552, 20872, 21508, and 22122) and NS5 found the best solutions in the first and second instances of the problem (21331, 22545, 21695, 21916, and 22626). Both NS4 and NS5 solutions are close to the optimum solutions for each instance. NS3 found the third best solutions for the problem (23075, 24182, 21101, 22142, and 22944), but there is already substantial difference between the NS3 solutions and the solutions from NS4 and NS5 in most instances. NS7 and NS8 solutions are almost identical, but nevertheless NS7 performs better than NS8 in three instances (24070, 24991, 24140, 24321, and 24794) while NS7 performed better than NS8 in the first and second instances (23922, 24479, 24168, 24350, and 24827). NS6 is not distant from NS7 or NS8 (25038, 25730, 25526, 24587, and 25662). NS2 is worse than NS6, NS7, and NS8 for all instances (25527, 27718, 26510, 26357, and 29442), but the differences are not substantial if we take into account the standard deviation in the performance of NS6, NS7, and NS8. NS1 found the worst solutions (82904, 93437, 93405, 85181, and 96965) and also appeared to have more variability across the instances. In SA with 10 seconds, NS3, NS4, NS5, NS6, NS7, and NS8 performed almost equally, which can be examined in Figure 4. NS4 found the best solutions for four of the five instances (21353, 22284, 20812, 21398, and 22216) and NS5 found the best solution in the fourth instance (21367, 22404, 20852, 21346, and 22304). NS3 (21603, 22567, 21000, 21573, and 22241), NS6 (21357, 22888, 21007, 22128, and 23078), NS7 (21368, 22454, 20984, 21642, and 22500), and NS8 (21601, 22535, 20905, 21700, and 22717) are similar and nearly identical to the solutions of NS4 and NS5. NS2 improved its performances considerably, but the difference in the performance of NS2 (24031, 24566, 22888, 22814, and 24311) and NS3-NS8 is more noticeable. Finally, NS1 (77805, 82614, 79205, 73869, and 89479) continued to perform much worse than all the other NS in all instances of the problem.

All the NS performed better with the 10-second limit and, other than NS4 and NS5 that found near-optimal solutions with the 1 second limit, all NS appeared to improve almost identically. NS1-NS4 intense searches do not appear to improve less than NS5-NS8, with more diverse search procedures, even with the increase of the computational time.

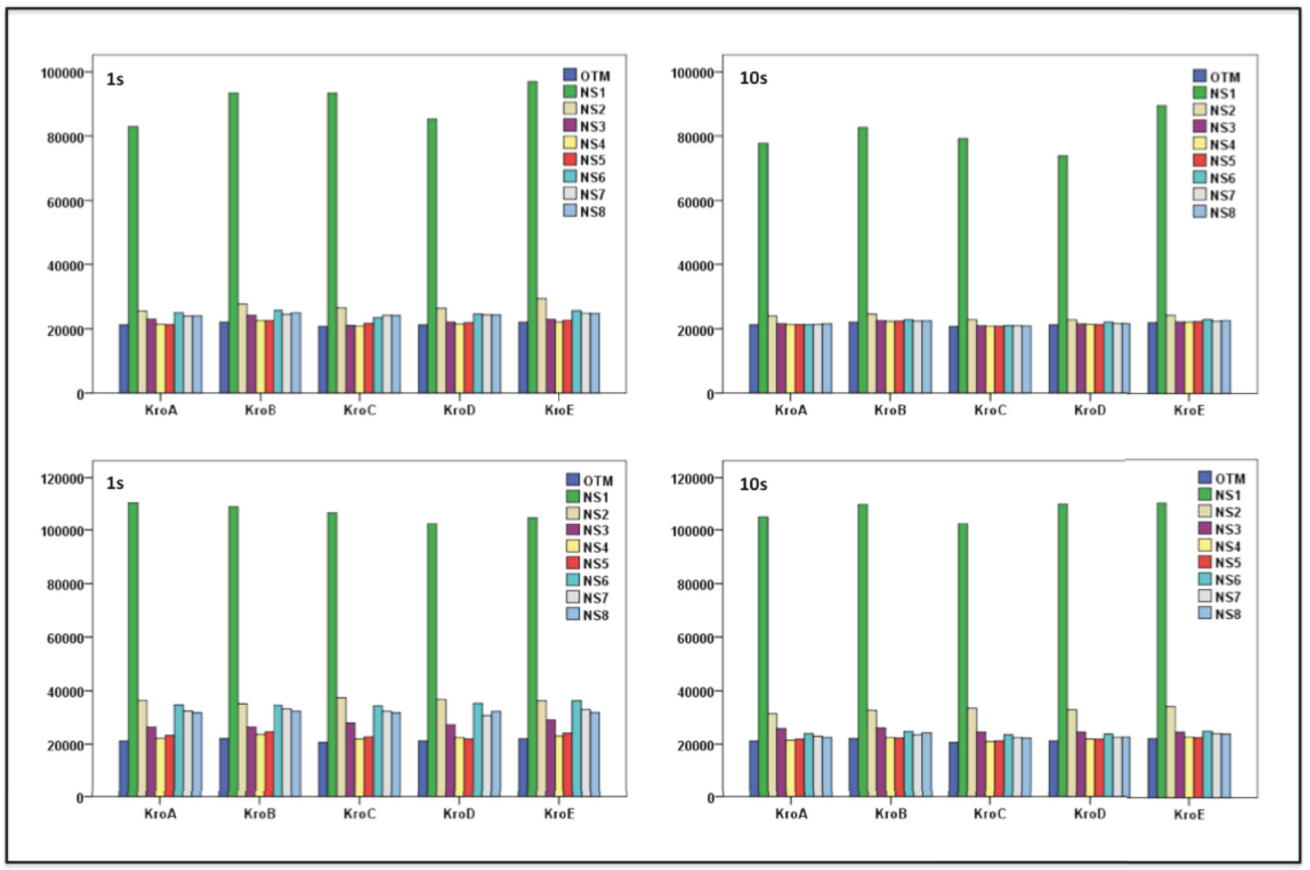


Figure 4: Results of SA and DABC.

In DABC with 1 second, NS4 found the best solutions in four of the five instances (22225, 23650, 21981, 22552, and 23161) and NS5 found the best solution in the fourth instance of the problem (23322, 24709, 22808, 21929, and 24165). SA performed better than DABC with both NS4 and NS5 with 1 second, but nevertheless the solutions are still close to the optimum solutions for each instance. NS3 found the third best solutions for all the instances of the problem (26340, 26384, 27983, 27339, and 29070), but the distances from the solutions found by NS4 and NS5 are already substantial. NS7 and NS8 solutions are almost identical, but NS8 performs better than NS7 in four instances (31739, 32341, 31690, 32302, and 31868) while NS7 performed better in the fourth instance (32464, 33186, 32353, 30746, and 33012). NS6 performs better than NS2 in three instances (34693, 35482, 34406, 35290, and 36253) while NS7 performed better in the second and fifth instances (36383, 35157, 37416, 36775, and 36202). NS1 found the worst solutions in all instances (110578, 109192, 106903, 102844, and 105054) and also appeared to be more inconsistent than all the other NS. In DABC with 10 seconds all NS solutions improved notably, but NS4 and NS5 continued to perform better. Nevertheless, the more pronounced improvement of NS7 and NS8 resulted in solutions almost identical to NS4 and NS5 solutions, which can be examined in Figure 4. NS5 found the best solutions for three of the five instances (21894, 22394, 21305, 21897, and 22511) and NS4 found the best solutions in two (21504, 22544, 21089, 21938, and 22654). NS4 and NS5 are followed by NS8 (22586, 24332, 22335, 22712, and 23772) and NS7 (23075, 23570, 22555, 22625, and 23953). NS3 and NS6 also improved their performances considerably, and overall the solutions are similar to the solutions of NS4, NS5, NS7, and NS8. NS6 (23989, 24814, 23624, 23793, and 24834) performed better than NS3 (25897, 26107, 24505, 24507, and 24461). NS2 improvement is not as noticeable in DABC as it was in SA, and the difference in performance between NS2 (31445, 32701, 33473, 32953, and 34060) and NS3-NS8 is self-evident. Finally, NS1 solutions did not improve with the increase of the duration. NS1 (105348, 110035, 102856, 110118, and 110077) continued to perform much worse than all the other NS in all instances of the problem.

Once more, all the NS improved their performance in the 10 seconds limit, but in opposition to the SA, the difference between NS1-NS4 more intense neighborhood structures and the NS5-NS8 more diverse neighborhood structures is noticeable. NS5-NS8 improved the solutions more than NS1-NS4, particularly for NS1, which did not appear to improve at all.

Overall SA performed better than the DABC, which is much more noticeable in the 1-second experiment. In the 10-second experiment the performance of both metaheuristics improved, but it appears to be a difference in improvement

Table 2: Normalized results of SA and DABC.

| | | | NS1 | NS2 | NS3 | NS4 | NS5 | NS6 | NS7 | NS8 |
|---|---|---|---|---|---|---|---|---|---|---|
| SA | 1s | KroA | 2.895 | 0.199 | 0.084 | 0.008 | 0.002 | 0.176 | 0.124 | 0.131 |
| | | KroB | 3.220 | 0.252 | 0.092 | 0.019 | 0.018 | 0.162 | 0.106 | 0.129 |
| | | KroC | 3.502 | 0.278 | 0.017 | 0.006 | 0.046 | 0.230 | 0.165 | 0.163 |
| | | KroD | 3.000 | 0.238 | 0.040 | 0.010 | 0.029 | 0.155 | 0.144 | 0.142 |
| | | KroE | 3.394 | 0.334 | 0.040 | 0.002 | 0.025 | 0.163 | 0.125 | 0.124 |
| SA | 10s | KroA | 2.656 | 0.129 | 0.015 | 0.003 | 0.004 | 0.004 | 0.004 | 0.015 |
| | | KroB | 2.731 | 0.110 | 0.019 | 0.006 | 0.012 | 0.034 | 0.014 | 0.018 |
| | | KroC | 2.817 | 0.103 | 0.012 | 0.003 | 0.005 | 0.012 | 0.011 | 0.008 |
| | | KroD | 2.469 | 0.071 | 0.013 | 0.005 | 0.002 | 0.039 | 0.016 | 0.019 |
| | | KroE | 3.055 | 0.102 | 0.008 | 0.007 | 0.011 | 0.046 | 0.020 | 0.029 |
| DABC | 1s | KroA | 4.196 | 0.710 | 0.238 | 0.044 | 0.096 | 0.630 | 0.525 | 0.491 |
| | | KroB | 3.932 | 0.588 | 0.192 | 0.068 | 0.116 | 0.603 | 0.499 | 0.461 |
| | | KroC | 4.152 | 0.803 | 0.349 | 0.059 | 0.099 | 0.658 | 0.559 | 0.527 |
| | | KroD | 3.830 | 0.727 | 0.284 | 0.059 | 0.030 | 0.657 | 0.444 | 0.517 |
| | | KroE | 3.760 | 0.640 | 0.317 | 0.050 | 0.095 | 0.643 | 0.496 | 0.444 |
| DABC | 10s | KroA | 3.950 | 0.478 | 0.217 | 0.010 | 0.029 | 0.127 | 0.084 | 0.061 |
| | | KroB | 3.970 | 0.477 | 0.179 | 0.018 | 0.011 | 0.121 | 0.065 | 0.099 |
| | | KroC | 3.957 | 0.613 | 0.181 | 0.016 | 0.027 | 0.139 | 0.087 | 0.076 |
| | | KroD | 4.171 | 0.548 | 0.151 | 0.030 | 0.028 | 0.117 | 0.063 | 0.067 |
| | | KroE | 3.988 | 0.543 | 0.108 | 0.027 | 0.020 | 0.125 | 0.085 | 0.077 |

of NS1-NS4 and NS5-NS8, which is much more noticeable in DABC than in SA. In DABC NS1-NS4 solutions did not improve as much as the most diverse NS, which becomes obvious once Figure 4 is closely examined.

*4.3. Statistical Results.* In the computational trials the difference between the evolution in the performance of the metaheuristics with the eight NS appears evident; however, to assess the implications of these apparent differences it is indispensable to examine the computational results in detail.

In order to examine the evolution of performance of each NS the computational results need to be normalized. Since the optimal solutions for each instance (KroA, KroB, KroC, KroD, and KroE) are available in TSPLIB [52], the results will be normalized into the mean percent deviation from the optimal solution, which is the best know metric to compare the performance of metaheuristics [53] or, in this case, the performance of the same two metaheuristics with different parameters. Once the results have been normalized, the improvement of the performance of the metaheuristics with each of the NS should become obvious and allow the improvement of the performance through each NS to be measured.

SA and DABC normalized results are presented in Table 2. It becomes clear that both metaheuristics improved their solutions with the increase of the available computational time; however, these improvements are not equal in both metaheuristics and all the NS examined. If the NS are divided between NS1, NS2, NS3, and NS4 that result in more intense and less disruptive searches and NS5, NS6, NS7, and NS8 which result in more diverse and disruptive searches the differences are obvious, particularly in DABC. In SA the difference in the improvement of the solutions between NS1-NS4 and NS5-NS8 is not evident.

In Figure 5 the comparison of the solutions for SA with all the NS for both 1- and 10-second experiments can be observed. SA solutions appear to scale consistently with the increase of the available time with all NS, even NS1, which performed inadequately in all the instances of the problem, improved with the increase of the computational time.

In Figure 6 the comparison of the solutions for DABC with all the NS for both 1 and 10 seconds can be analyzed. In DABC NS5-NS8 appear to improve the performance of the metaheuristic more than the NS1-NS4. Moreover, NS1 worsen the performance of DABC with the increase of the computational time, which does not happen in SA.

Table 3 presents the improvement in the mean percent deviation from the optimum. Overall, the improvement of NS1-NS4 is less noticeable than the improvements of NS5-NS8. In SA the difference between NS1-NS4 and NS5-NS8 is not as prominent as in DABC. In SA, NS1-NS4 had a mean improvement of 16.5% and NS5-NS8 of 10.2%, while in DABC, the NS1-NS4 improvement is of 6.8% and NS5-NS8 of 35.4%. For instance, NS1 in DABC did not improve the solutions in three of the five instances of the problem.

In Figure 7 the boxplot of the mean improvement can be verified. NS1-NS4 resulted in a mean improvement of 11,6% while the mean improvement of NS5-NS8 was of 22,8%. The standard deviation of NS1-NS4 is of 18,1%, which is almost identical to the standard deviation of NS5-NS8, which is 18,3%. However NS1-NS4 result in a much wider range than NS5-NS8, which can be explained by the performance of DABC with NS1-NS4.

Table 3: Improvement of NS1-NS4 and NS5-NS8.

| | | NS1 | NS2 | NS3 | NS4 | NS5 | NS6 | NS7 | NS8 |
|---|---|---|---|---|---|---|---|---|---|
| **SA** | KroA | 0.240 | 0.070 | 0.069 | 0.005 | -0.002 | 0.173 | 0.120 | 0.116 |
| | KroB | 0.489 | 0.142 | 0.073 | 0.012 | 0.006 | 0.128 | 0.091 | 0.111 |
| | KroC | 0.684 | 0.175 | 0.005 | 0.003 | 0.041 | 0.218 | 0.153 | 0.156 |
| | KroD | 0.531 | 0.166 | 0.027 | 0.005 | 0.027 | 0.115 | 0.127 | 0.123 |
| | KroE | 0.339 | 0.233 | 0.032 | -0.004 | 0.015 | 0.117 | 0.105 | 0.094 |
| **DABC** | KroA | 0.246 | 0.232 | 0.021 | 0.034 | 0.067 | 0.503 | 0.441 | 0.430 |
| | KroB | -0.038 | 0.111 | 0.013 | 0.050 | 0.105 | 0.482 | 0.434 | 0.362 |
| | KroC | 0.195 | 0.190 | 0.168 | 0.043 | 0.072 | 0.520 | 0.472 | 0.451 |
| | KroD | -0.342 | 0.179 | 0.133 | 0.029 | 0.002 | 0.540 | 0.381 | 0.450 |
| | KroE | -0.228 | 0.097 | 0.209 | 0.023 | 0.075 | 0.517 | 0.411 | 0.367 |

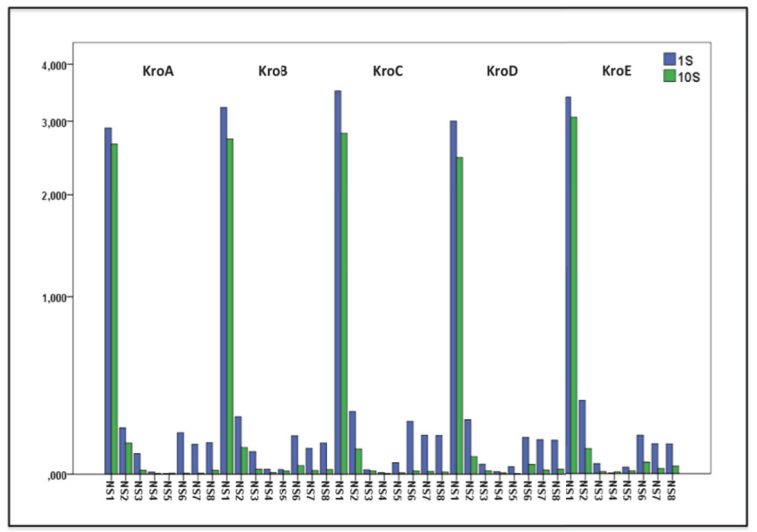


Figure 5: Comparison of the 1 and 10 results for SA.

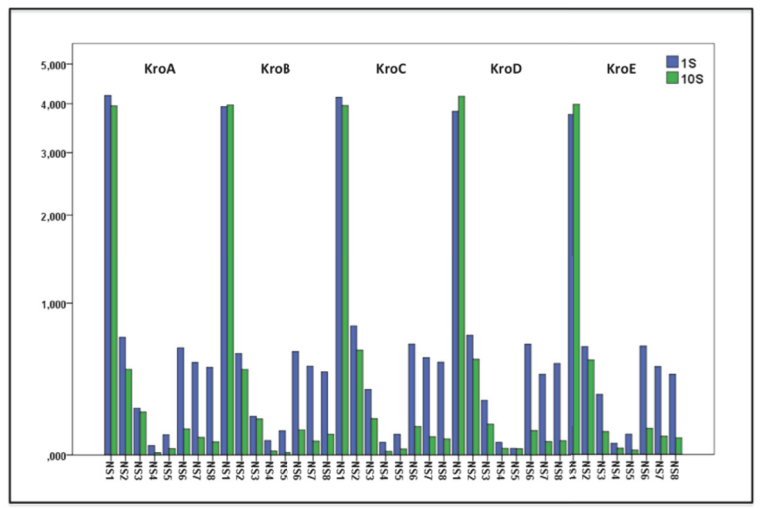


Figure 6: Comparison of the 1 and 10 results for DABC.

Table 4: Wilcoxon Mann-Whitney *U* Test.

| Parameter | SA | DABC |
|---|---|---|
| Mann-Whitney U | 198,500 | 56,000 |
| Wilcoxon W | 408,500 | 266,000 |
| Z | -0,041 | -3,895 |
| Asymp. Sig. (2-tailed) | 0,968 | 0,000 |
| Exact Sig. [2∗(1- tailed Sig.)] | 0,968 | 0,000 |
| Exact Sig. (2-tailed) | 0,974 | 0,000 |
| Exact Sig. (1-tailed) | 0,487 | 0,000 |

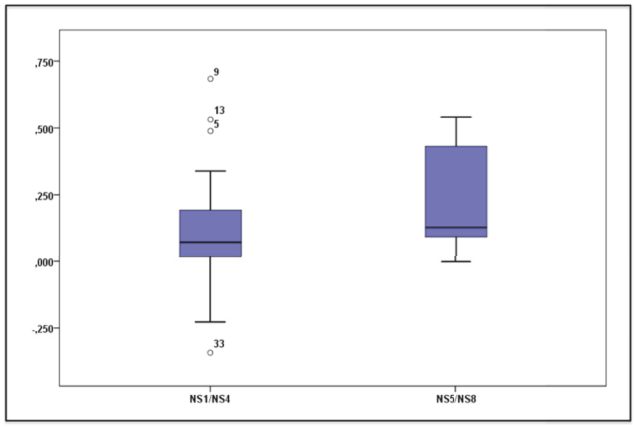


Figure 7: Boxplot of NS1-NS4 and NS4-NS8.

All the previous results appear to demonstrate that NS5-NS8 improved the solutions more than NS1-NS4, but these differences require validation. Since the solutions cannot be presumed to follow a normal distribution, the differences in median between NS1-NS4 and NS5-NS8 will be examined with a nonparametric test, in this case the Wilcoxon Mann-Whitney *U* test. Moreover, Figures 5 and 6 show the SA performance scales uniformly with all NS, which does not happen with DABC. In order to corroborate this supposition the Wilcoxon Mann-Whitney *U* test examined SA and DABC separately with the previously identified hypothesis:

$$H_0: \eta_{NS1/4} = \eta_{NS5/8}.$$
$$H_1: \eta_{NS1/4} \neq \eta_{NS5/8}.$$

Table 4 presents the results of the Wilcoxon Mann-Whitney *U* test. For DABC the Wilcoxon Mann-Whitney *U* test significance level of 0.05 rejected $H_0$ (0.000), which validated the apparent discrepancy between the median of the improvement with NS1-NS4 and NS5-NS8. It indicates that the more disruptive structures (NS5-NS8) outperform the more intense structures (NS1-NS4) when the computational time increases. For SA the Wilcoxon Mann-Whitney *U* test significance level of 0.05 could not reject $H_0$ (0.974), which validates the apparent uniform improvement of the solutions with all NS.

Overall the computational results substantiated the results presented in [40], but with the inclusion of four more NS, the disparities between SA and DABC became much more evident. DABC appears to be more sensitive to the NS than SA, whose performance appeared to scale uniformly with all NS, which corroborate Park and Kim in [41], who state that SA do not require a detailed parameterization procedure to perform well, if there is enough computational time and a high enough initial temperature.

In SA the selected NS did not overly impact the performance of the MH, which explains the lack of variance in the improvements of the solutions with NS1-NS4 and NS5-NS8, with the increase of the computational time. On the other hand, in DABC the results demonstrated that NS5-NS8 improved the solutions more than NS1-NS4 with the increase of the computational time, which was demonstrated through the Wilcoxon Mann-Whitney *U* test. Unlike SA, DABC requires a detailed relation between the NS and the other parameters to perform satisfactorily.

In summary, through the analysis carried out in this work, it was demonstrated that while the NS is a particular parameter, which is common to all metaheuristics, and even other search techniques, it should be treated as another parameter, while contributing to the overall balance of how intense/diverse a search procedure is.

Studies that demonstrate how appropriate one NS is for a specific problem should consider how the NS interacts with the other parameters and other limitations of the optimization procedure, such as how the NS performs once the MH has a time constraint, and this issue was focused in this paper.

## 5. Conclusion

Through the computational study it was demonstrated that the performance of each NS in the TSP problem is related to the other parameters of the search procedure. Once the computational time was increased from 1 second to 10 seconds the more intense NSs became less competitive than the more disruptive NSs. In all the conducted tests the improvement of the more intense NSs was less prominent than that of the more diverse NSs. For DABC the Wilcoxon Mann-Whitney test demonstrated the statistical difference in the improvement of NS1-NS4 and NS5-NS8 (0.000), with much more noticeable improvement of the more diverse NSs. For SA the improvement of the solutions appeared more uniform with an almost nonexistent variance in improvement, which was validated with the Wilcoxon Mann-Whitney (0.974). Since SA does not require comprehensive parameterization, the performance of this MH appeared to scale consistently with the increase of the available computational time with all NSs. On the other hand, DABC requires a meticulous definition of the parameters and as such the relation between the NSs and the other parameters needs to be carefully determined for the MH to perform well within the available execution time. For both MH, but particularly for DABC, which requires a meticulous parameterization procedure, the NS should not be determined especially for the problem; it should take into account the relation between the NS and other parameters of the MH.

Further work should focus on a more in-depth computational study, in order to validate the results of this document.

Additional optimization problems with specific NS should also be further examined, for instance, for solving other kind of COPs, such as scheduling problems, occurring in different manufacturing environments, and an extra MH should be tested to evaluate how each MH is sensitive to the NS. More NSs should also be used, to examine their behavior with different time constraints.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work is supported by FEDER Funds through the "Programa Operacional Factores de Competitividade (COMPETE)" and by National Funds through FCT "Fundação para a Ciência e a Tecnologia" under the Projects FCOMP-01-0124-FEDER-PEst-OE/EEI/UI0760/2011, PEst-OE/EEI/UI0760/2014, and PEst2015-2020.

## References

[1] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer, New York, NY, USA, 4th edition, 2012.

[2] V. K. Manupati, M. G. Krishnan, M. L. R. Varela, and J. Machado, "Telefacturing Based Distributed Manufacturing Environment for Optimal Manufacturing Service by Enhancing the Interoperability in the Hubs," *Journal of Engineering (United States)*, vol. 2017, 15 pages, 2017.

[3] C.-S. Jeong and M.-H. Kim, "Fast parallel simulated annealing for traveling salesman problem on SIMD machines with linear interconnections," *Parallel Computing. Systems & Applications*, vol. 17, no. 2-3, pp. 221–228, 1991.

[4] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, pp. 791–812, 1958.

[5] J.-C. Créput and A. Koukam, "A memetic neural network for the Euclidean traveling salesman problem," *Neurocomputing*, vol. 72, no. 4–6, pp. 1250–1264, 2009.

[6] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.

[7] S. Lin, "Computer solutions of the traveling salesman problem," *Bell Labs Technical Journal*, vol. 44, pp. 2245–2269, 1965.

[8] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and Q. X. Wang, "Particle swarm optimization-based algorithms for TSP and generalized TSP," *Information Processing Letters*, vol. 103, no. 5, pp. 169–176, 2007.

[9] Y. Chen and P. Zhang, "Optimized annealing of traveling salesman problem from the nth-nearest-neighbor distribution," *Physica A: Statistical Mechanics and its Applications*, vol. 371, no. 2, pp. 627–632, 2006.

[10] J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem," *Operations Research*, vol. 11, no. 6, pp. 972–989, 1963.

[11] F. Glover, *Tabu Search: A Tutorial*, Center for Applied Artificial Intelligence, University of Colorado, 1990.

[12] Y. Marinakis, M. Marinaki, and G. Dounias, "Honey bees mating optimization algorithm for the Euclidean traveling salesman problem," *Information Sciences*, vol. 181, no. 20, pp. 4684–4698, 2011.

[13] Y.-T. Wang, J.-Q. Li, K.-Z. Gao, and Q.-K. Pan, "Memetic algorithm based on improved Inver-over operator and Lin-Kernighan local search for the Euclidean traveling salesman problem," *Computers & Mathematics with Applications*, vol. 62, no. 7, pp. 2743–2754, 2011.

[14] D. Bonachea, E. Ingerman, J. Levy, and S. Mcpeak, "An Improved Adaptive Multi-Start Approach To Finding Near-Optimal Solutions To The Euclidean TSP," in *Proceedings of the Genetic And Evolutionary Computation Conference*, pp. 143–150, 2000.

[15] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[16] A. J. Orman and H. P. Williams, "A survey of different integer programming formulations of the travelling salesman problem," *Optimisation Econometric and Financial Analysis, Advances in Computational Management Science*, vol. 9, pp. 91–104, 2007.

[17] G. Laporte, "The traveling salesman problem: an overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

[18] R. Matai, S. P. Singh, and M. L. Mittal, "Traveling salesman problem: an overview of applications, formulations, and solution approaches," in *Traveling Salesman Problem: Theory And Applications*, D. Davendra, Ed., Chapter 1, pp. 1–24, 2011.

[19] L. R. Varela and R. A. Ribeiro, "Evaluation of Simulated Annealing to solve fuzzy optimization problems," *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, vol. 14, no. 2, pp. 59–71, 2003.

[20] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2003.

[21] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. M. Zaidi, "The bees algorithm," Technical Note, Cardiff University, 2005.

[22] E. G. Talbi, *Meta-heuristics: From Design to Implementation*, John Wiley & Sons, 2009.

[23] T. Ghosh, S. Sengupta, M. Chattopadhyay, and P. K. Dan, "Meta-heuristics in cellular manufacturing: a state-of-the-art review," *International Journal of Industrial Engineering Computations*, vol. 2, no. 1, pp. 87–122, 2011.

[24] I. H. Osman and J. P. Kelly, *Meta-Heuristics: An Overview. Meta-Heuristics Theory and Applications*, Kluwer Academic Publishers, 1996.

[25] I. Pereira, A. Madureira, P. B. de Moura Oliveira, and A. Abraham, "Tuning meta-heuristics using multi-agent learning in a scheduling system," in *Transactions on Computational Science*, vol. 8160, pp. 190–210, Springer, 2013.

[26] A. Shrestha and A. Mahmood, "Improving genetic algorithm with fine-tuned crossover and scaled architecture," *Journal of Mathematics*, Art. ID 4015845, 10 pages, 2016.

[27] I. Boussai, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[28] S. Mishra and R. Srivastava, "Fuzzy soft compact topological spaces," *Journal of Mathematics*, Art. ID 2480842, 7 pages, 2016.

[29] A. Nagoor Gani, K. Kannan, and A. R. Manikandan, "Inner product over fuzzy matrices," *Journal of Mathematics*, Art. ID 6521893, 10 pages, 2016.

[30] T. Kumar and R. K. Bajaj, "On complex intuitionistic fuzzy soft sets with distance measures and entropies," *Journal of Mathematics*, Art. ID 972198, 12 pages, 2014.

[31] S. Kirkpatrick, J. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *American Association for the Advancement of Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[32] V. Černý, "Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.

[33] D. Karaboga and B. Gorkemli, "A combinatorial Artificial Bee Colony algorithm for traveling salesman problem," in *Proceedings of the International Symposium on Innovations in Intelligent Systems and Applications (INISTA '11)*, pp. 50–53, Istanbul, Turky, June 2011.

[34] S. S. Choong, L. Wong, and C. P. Lim, "An artificial bee colony algorithm with a modified choice function for the Traveling Salesman Problem," in *Proceedings of the 2017 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 357–362, Banff, AB, October 2017.

[35] S. Sundar, P. N. Suganthan, C. T. Jin, C. T. Xiang, and C. C. Soon, "A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint," *Soft Computing*, vol. 21, no. 5, pp. 1193–1202, 2017.

[36] R. Soto, B. Crawford, L. Vásquez et al., "Solving the Manufacturing Cell Design Problem Using the Artificial Bee Colony Algorithm," in *Multi-disciplinary Trends in Artificial Intelligence*, vol. 10607 of *Lecture Notes in Computer Science*, pp. 473–484, Springer International Publishing, Cham, 2017.

[37] A. S. Santos, A. M. Madureira, and M. R. Varela, "Evaluation of the simulated annealing and the discrete artificial bee colony in the weight tardiness problem with taguchi experiments parameterization," *Advances in Intelligent Systems and Computing*, vol. 557, pp. 718–727, 2017.

[38] B. Adenso-Díaz and M. Laguna, "Fine-tuning of algorithms using fractional experimental designs and local search," *Operations Research*, vol. 54, no. 1, pp. 99–114, 2006.

[39] F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy, "An experimental investigation of model-based parameter optimisation: SPO and beyond," in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference, GECCO '09*, pp. 271–278, July 2009.

[40] A. S. Santos, A. M. Madureira, and M. L. R. Varela, "Study on the impact of the NS in the performance of meta-heuristics in the TSP," in *Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016*, pp. 1110–1115, Budapest, Hungary, October 2016.

[41] M.-W. Park and Y.-D. Kim, "A systematic procedure for setting parameters in simulated annealing algorithms," *Computers & Operations Research*, vol. 25, no. 3, pp. 207–217, 1998.

[42] S. Anily and A. Federgruen, "Simulated annealing methods with general acceptance probabilities," *Journal of Applied Probability*, vol. 24, no. 3, pp. 657–667, 1987.

[43] R. W. Eglese, "Simulated annealing: a tool for operational research," *European Journal of Operational Research*, vol. 46, no. 3, pp. 271–281, 1990.

[44] J. Rose, W. Klebsch, and J. wolf, "Temperature Measurement and Equilibrium Dynamics of Simulated Annealing Placements," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 3, pp. 253–259, 1990.

[45] B. Akay and D. Karaboga, "Parameter tuning for the artificial bee colony algorithm," in *Computational Collective Intelligence, Semantic Web, Social Network and Multiagent Systems*, vol. 5796 of *Lecture Notes in Computer Science*, pp. 608–619, 2009.

[46] S. Kockanat and N. Karaboga, "Parameter tuning of artificial bee colony algorithm for Gaussian noise elimination on digital images," in *Proceedings of the 2013 IEEE International Symposium on Innovations in Intelligent Systems and Applications, IEEE INISTA 2013*, June 2013.

[47] M. S. Kiran and M. Gündüz, "The Analysis of Peculiar Control Parameters of Artificial Bee Colony Algorithm on the Numerical Optimization Problems," *Journal of Computer and Communications*, vol. 2, no. 4, pp. 127–136, 2014.

[48] Y.-F. Liu and S.-Y. Liu, "A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem," *Applied Soft Computing*, vol. 13, no. 3, pp. 1459–1463, 2013.

[49] M. D. Besten and T. Stützle, "Neighborhoods Revisited: an Experimental Investigation into the effectiveness of variable neighborhood descent for scheduling," in *Proceedings of the 4th Metaheuristics International Conference*, pp. 545–549, 2001.

[50] M. Sevkli and M. Emin Aydin, "Variable Neighborhood Search for Job Shop Scheduling Problems," *Journal of Software*, vol. 1, no. 2, pp. 34–39, 2006.

[51] R. K. Ahuja, O. Ergun, J. B. Orlin, and A. P. Punnen, "A survey of very large-scale neighborhood search techniques," *Discrete Applied Mathematics: The Journal of Combinatorial Algorithms, Informatics and Computational Sciences*, vol. 123, no. 1-3, pp. 75–102, 2002.

[52] G. Reinelt, "TSPLIB: a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[53] J. Silberholz and B. Golden, "Comparison of Metaheuristics," in *Handbook of Metaheuristics*, vol. 146 of *International Series in Operations Research & Management Science*, pp. 625–640, Springer US, Boston, MA, 2010.