Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 3046830, 20 pages
https://doi.org/10.1155/2017/3046830

*Review Article*

# Bee-Inspired Algorithms Applied to Vehicle Routing Problems: A Survey and a Proposal

**Thiago A. S. Masutti and Leandro N. de Castro**

*Natural Computing and Machine Learning Laboratory (LCoN), Graduate Program in Electrical Engineering and Computing, Mackenzie Presbyterian University, R. da Consolação 930, Higienópolis, 01302-000 São Paulo, SP, Brazil*

Correspondence should be addressed to Leandro N. de Castro; lnunes@mackenzie.br

Received 25 January 2017; Revised 29 July 2017; Accepted 22 August 2017; Published 8 October 2017

Academic Editor: Jorge Magalhaes-Mendes

Copyright © 2017 Thiago A. S. Masutti and Leandro N. de Castro. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Vehicle routing problems constitute a class of combinatorial optimization tasks that search for optimal routes (e.g., minimal cost routes) for one or more vehicles to attend a set of nodes (e.g., cities or customers). Finding the optimal solution to vehicle routing tasks is an NP-hard problem, meaning that the size of problems that can be solved by exhaustive search is limited. From a practical perspective, this class of problems has a wide and important set of applications, from the distribution of goods to the integrated chip design. Rooted on the use of collective intelligence, swarm-inspired algorithms, more specifically bee-inspired approaches, have been used with good performance to solve such problems. In this context, the present paper provides a broad review on the use of bee-inspired methods for solving vehicle routing problems, introduces a new approach to solve one of the main tasks in this area (the travelling salesman problem), and describes open problems in the field.

## 1. Introduction

The characteristics of an optimization problem, such as the high number of possible solutions (size of the search space) and number and type of constraints, can prevent its solution by exhaustive search methods in a feasible time, even if large amounts of computational resources are employed. For this reason, instead of using methods that calculate the exact solution to the problem, techniques that propose candidate solutions and consider the history of results in the search process, modifying these solutions iteratively until a satisfactory solution is found, are usually employed. These methods exchange the guarantee of the exact solution for a satisfactory solution that can be obtained with an acceptable computational cost. These methods are normally called metaheuristics [1–3].

The planning of routes used to transport products is one of the ways to improve the use of resources in various transportation modals. For instance, a beverage distributor has its vehicle filled with goods and must distribute them to a certain group of customers. Defining the best sequence of customer visits, known as a route, can help reduce the cost of this operation and, consequently, impact the final product cost. In the literature, the route planning task is defined as a vehicle routing problem (VRP), with the travelling salesman problem (TSP) as one of the most elementary and widely studied cases [4–10]. The VRPs belong to the category of NP-hard problems, making them difficult to solve by exact methods, mainly for larger instances. Therefore, much of the study of this class of problems is based on metaheuristics [11, 12].

One area that has received considerable attention from the scientific community over the past decades is that of nature-inspired algorithms [13–15]. Studies in this area seek inspiration in biological phenomena for the development of algorithms capable of solving problems that are not satisfactorily solved by traditional techniques [16]. In such cases, heuristics inspired by insect behaviors have shown good results. Swarm intelligence (SI) is a term used to describe algorithms that have as inspiration the collective behavior of social insects or other types of social animals [17, 18]. Within swarm intelligence, the agents in the swarm

act with no supervision, being affected by what happens in the surroundings, interacting with the environment and with other agents. An important characteristic of SI systems is the self-organization: with the low-level interactions within the agents, the swarm is capable of providing global responses. Among the most studied algorithms in this area one can mention the Ant Colony Optimization (ACO) [19] and the Particle Swarm Optimization (PSO) [20].

Some social species of bees clearly present characteristics and principles related to swarm intelligence, making them a good inspiration to optimization algorithms. Some of the main collective behaviors of bee colonies during foraging that can be highlighted as inspiration for the design of algorithms are as follows [21]: (1) bees dance to recruit nestmates to a food source; (2) bees adjust the exploration and recovery of food according to the colony state; (3) bees exploit multiple food sources simultaneously; (4) there is a positive linear relationship between the number of bees dancing and the number of bees recruited; (5) recruitment continues until a threshold number of bees is reached; (6) the quality of the food source influences the bee dance; and (7) all bees retire at some point in time, meaning that bees stop recruiting other bees. For generic reviews of bee-inspired algorithms and applications, the reader is invited to refer to Karaboga and Akay [22], Bitam et al. [23], Ruiz-Vanoye et al. [24], Verma and Kumar [25], Karaboga et al. [26], and Agarwal et al. [27].

This paper starts by bringing a chronological review on bee-inspired algorithms applied to vehicle routing problems. The review includes a brief description of the key biological mechanisms of the bee metaphor and a mathematical description of vehicle routing problems. It then follows with a taxonomy of bee-inspired algorithms and a description of the four standard approaches, presenting the main types of (bee) agents in the metaphor, a simplified pseudocode of the algorithm, and a brief description of its main steps. The review of the papers focuses on the problem solved, the types of bee agents or base algorithm used, how the algorithms were assessed, and comments on their overall performance.

The last part of the paper describes a new bee-inspired proposal, named TSPoptBees, originally designed to solve continuous optimization problems [21] and then adapted to solve VRPs, more specifically the travelling salesman problem. The proposed algorithm is applied to 28 TSP instances and its results are compared to the best-known solutions from the literature and with the results of many other bee-inspired algorithms cited in the review part of the paper. The paper is concluded with general comments and a proposal of how to extend the algorithm to the other vehicle routing problems.

The remainder of this paper is organized as follows. Section 2 provides a brief introduction to bee colonies and vehicle routing problems; Section 3 describes the base algorithms used in the literature to solve vehicle routing problems; and Section 4 briefly reviews each work found in the literature related to bee-inspired algorithms to solve vehicle routing problems. The proposed algorithm is detailed in Section 5 and its performance is assessed in Section 6. The paper is concluded in Section 7 with general comments and perspectives for future research for the area.

## 2. Fundamentals of Bee Colonies and Vehicle Routing Problems

This section provides a brief introduction to the use of bee colonies as metaphors for swarm intelligence and a standardized mathematical description of vehicle routing problems, to be used over the whole paper.

*2.1. Bee Colonies as Metaphors for Swarm Intelligence.* A notable feature of swarm intelligence is self-organization, whereby the system is able to present responses at a global level through the low-level iterations among the agents themselves and the environment. Bonabeau et al. [17] highlight four characteristics of self-organization in swarms: positive feedback, through simple behavioral rules that promote the creation of appropriate structures; negative feedback, which acts as a counterbalance to positive feedback; oscillations, such as random behaviors, errors, and task switching; and multiple interactions among the agents and the environment, allowing the exchange of information. Millonas [28] highlighted five principles for a swarm to present intelligent behavior: proximity (individuals should be able to perform simple tasks); quality (the swarm must be sensitive to quality factors); diversity (the swarm should not allocate all its resources in a single medium, but distribute them); stability (individuals should not change their behavior in response to all changes in the environment); and adaptability (the swarm must be able to change its behavior when necessary).

Some species of bees clearly present the characteristics described by Bonabeau et al. [17] and the principles defined by Millonas [28], which motivates the use of bee colonies as metaphors for the design of algorithms for solving complex problems. Karaboga and Akay [22] highlighted some tasks performed by bee colonies that are most used in the literature as metaphors for swarm intelligence, as well as the clear division of tasks between bee types:

(i) *Queen bee*: it is the only female of the colony that lays eggs, being the progenitor of all the other bees in the colony. It can live for several years, mating only once. Fertilization can occur for two or more years, using sperm stored during mating. When there is a lack of food sources, it produces more eggs and when the colony is very populated it stops producing them. After consuming the sperm in its spermatheca, it produces eggs that have not been fertilized and one of these descendants will become the new queen.

(ii) *Drones*: they are the male bees of the hive, with reproductive role, being considered the progenitors of the other bees of the colony. They are produced from unfertilized eggs and are fed differently when in the larval stage. Depending on the period there may be hundreds of drones in the colony; however they do not live more than six months. Their main task is to mate with the queen, dying after that.

(iii) *Workers*: they are responsible for various operational tasks of the hive, such as collecting and storing food, removing debris, and protecting the hive. They can

live for a few weeks or months and the tasks to which they are allocated depend on their age and the colony needs. In general, in the second half of their lives they forage for food.

(iv) *Mating flight*: the mating of the queen takes place in the air during the so-called mating flight. This starts with a dance performed by the queen and the drones follow her. The mating between the queen and a drone is probabilistic, according to the speed of the queen's flight and the fitness of the drone and queen. The drone sperm is stored in the spermatheca of the queen and can be used in the descendants fertilized by the queen.

(v) *Foraging*: this is one of the most important tasks for the hive. External factors (odor, location, and presence of other bees in the food source) and internal factors (souvenir and odor of the location) on bees influence this process. It begins with the worker leaving the hive in search of the food source. After finding this source, the nectar is collected and stored in its stomach, and after returning to the nest it deposits the nectar in the combs.

(vi) *Dancing*: dancing is the way bees inform others of good food sources. After unloading the nectar, the bee that found an attractive food source performs a series of movements, called waggle dance. Information such as quality, direction, and distance from the source of food is passed through the dance.

*2.2. Optimization and Vehicle Routing Problems.* Optimization consists of determining the values of a set of variables that minimize or maximize a given mathematical expression, satisfying all problem constraints [2, 3]. Perhaps the most intuitive way to solve a given optimization problem is to list all possible solutions, evaluate them, and use the best solution. However, this approach, known as brute force, is not efficient depending on the characteristics of the problem. The main drawback in using full enumeration is that it becomes computationally impractical depending on the number of possible solutions to the problem. This means that this exact solution approach would be valid only for simpler problems, which hardly occurs in practical applications.

An example of an optimization problem commonly used in the literature is the travelling salesman problem (TSP) [5, 7, 9]. In a simple way, TSP can be described as follows: given a set of cities, the salesman should visit every city once, coinciding the initial city with the final one, so that the cost of the path travelled is minimal. For the asymmetric TSP, in which the distance between a city A and another city B can be different from the distance between B and A, the number of possible solutions is $(n-1)!$, where $n$ is the number of cities. This means that the number of possible solutions has a factorial growth with the size of the problem.

The TSP can be mathematically described as follows. Given a set of $n$ cities and the cost $\mathbf{C}_{ij}$ $(i, j = 1, 2, 3, \ldots, n)$ of going from city $i$ to city $j$, the TSP aims at determining a permutation of $\boldsymbol{\pi}$ the cities that minimize

$$\sum_{i=1}^{n-1} \left(\mathbf{C}_{\boldsymbol{\pi}_i \boldsymbol{\pi}_{i+1}}\right) + \mathbf{C}_{\boldsymbol{\pi}_n \boldsymbol{\pi}_1}. \tag{1}$$

TSP represents one of the elementary vehicle routing problems and, despite its simple description, its exact solution becomes complex because of the computational cost required, being part of the class of NP-hard problems [29]. Due to its academic importance and wide application in practical problems, TSP has been receiving great attention for more than 60 years [5, 10]. As a result, well-consolidated TSP-oriented review works can be found in the literature [6, 9, 29], which present formulations, examples of practical applications, and classical solution algorithms.

Vehicle routing problems closer to practical applications appear with the addition of some constraints to the TSP. The Multiple Travelling Salesman Problem (MTSP) is a generalization of TSP in which more than one salesman is used in the solution and a common city, the depot, is used as starting and ending point by all the salesmen. Given a number $M$ of salesmen, a depot $i_d$, a set $\mathbf{S} = \{i_1, i_2, i_3, \ldots, i_n\}$ with $n$ intermediate cities, and the cost $\mathbf{C}_{ij}$ $(i, j = 1, 2, 3, \ldots, n+1)$ of going from one city to another and also from each city to the depot, the MTSP consists of determining the set of permutations $\boldsymbol{\pi}_m$ $(m = 1, 2, 3, \ldots, M)$ from the elements in $\mathbf{S}$ so as to minimize

$$\sum_{m=1}^{M} \left( \mathbf{C}_{i_d, \boldsymbol{\pi}_{m,1}} + \sum_{i=1}^{n_m - 1} \mathbf{C}_{\boldsymbol{\pi}_{m,i}, \boldsymbol{\pi}_{m,i+1}} + \mathbf{C}_{\boldsymbol{\pi}_{m,n_m}, i_d} \right), \tag{2}$$

where $\boldsymbol{\pi}_{m,i}$ represents the city visited in order $i$ in route $m$ and $n_m$ is the number of intermediary cities visited in route $m$. The MTSP already presents enough characteristics to represent practical problems, such as school vehicle routing [30]. Bektas [31] presented other examples of practical applications and algorithms for MTSP.

The addition of some constraints in MTSP leads to another classical problem, the capacitated vehicle routing problem (CVRP). The CVRP can be seen as an MTSP with two modifications: (1) each intermediate city represents a customer with a certain demand for a product and (2) each salesman represents a vehicle with a limited capacity. Given a number $K$ of identical vehicles, each with a capacity $Q$, a depot $i_d$, a set $S = \{i_1, i_2, i_3, \ldots, i_n\}$ with $n$ customers, each with a demand $q_i$ $(i = 1, 2, 3, \ldots, n)$, and the cost $\mathbf{C}_{ij}$ $(i, j = 1, 2, 3, \ldots, n+1)$ of going from customer $i$ to customer $j$ and from each customer to the depot, the CVRP consists of determining the permutation $\boldsymbol{\pi}$ of the elements in $S$ so as to minimize

$$\sum_{k=1}^{K} \left( \mathbf{C}_{i_d, \boldsymbol{\pi}_{k,1}} + \sum_{i=1}^{n_k - 1} \mathbf{C}_{\boldsymbol{\pi}_{k,i}, \boldsymbol{\pi}_{k,i+1}} + \mathbf{C}_{\boldsymbol{\pi}_{k,n_k}, i_d} \right), \tag{3}$$

subject to

$$\sum_{i=1}^{n_k} q_{\boldsymbol{\pi}_{k,i}} \leq Q \quad \forall k, \tag{4}$$

Table 1: Name of the base algorithm inspired by different behaviors of bees.

| Honeybees behavior | Algorithm name |
|---|---|
| Foraging | Artificial Bee Colony (ABC) |
| | Bee Colony Optimization (BCO) |
| | Bee System (BS) |
| Marriage | Marriage in Honeybees Optimization (MHBO) |

where $\pi_{k,i}$ represents the city visited in order $i$ in route $k$ and $n_k$ is the number of customers visited by vehicle $k$. For a review of practical applications and solution algorithms the works of Laporte et al. [11] and Laporte [10] are suggested.

## 3. Bee-Inspired Algorithms: A Taxonomy and Standard Approaches

In the literature, there are basically four different algorithms forming the basis of all bee-inspired approaches for solving *vehicle routing problems* (VRPs). Thus, the works to be reviewed here are either the proposal of one algorithm or a modification of it to solve a different problem or to improve its performance. The domain of bee-inspired algorithms is not constrained to the ones presented in this section, but these are the ones so far used to tackle vehicle routing problems. As shown in Table 1, these algorithms can be divided into two subclasses based on the bees' behavior used as inspiration [23]. Each of the base algorithms is described in the following subsections.

*3.1. Artificial Bee Colony.* The Artificial Bee Colony (ABC) algorithm was introduced by Karaboga [32] and later extended by Basturk and Karaboga [33] to solve numeric function optimization problems. This algorithm is inspired by the honeybees' foraging behavior. The first modification of the ABC to solve a vehicle routing problem was proposed by Banharnsakun et al. [34] and the problem chosen was the *travelling salesman problem* (TSP). This section describes the ABC algorithm as a general-purpose optimization model, as in Karaboga et al. [26]. The modifications proposed in each work to solve the vehicle routing problem are described in Section 4. The ABC algorithm works with three kinds of agents (bees):

(i) *Employed bees*: bees already associated with a food source, which represents a solution

(ii) *Onlooker bees*: bees that will observe the dance of employed bees and follow them to the neighborhood of their food sources

(iii) *Scout bees*: bees that wander around the search space to find new food sources; this can be a random or guided process.

Each of these agents is used during specific phases of the algorithm, as presented and summarized in Box 1.

```
Initialization
while stopping criterion is not met
  Employed bees phase
  Onlooker bees phase
  Scout bees phase
End
```

Box 1: Structure of the ABC algorithm and its main phases.

*3.1.1. Initialization.* The initial food sources are discovered by scout bees. The number of food sources is the same as the number of employed bees in the hive. Each food source is then associated with an employed bee.

*3.1.2. Employed Bees Phase.* During this phase, an employed bee will search for better food sources in its neighborhood. Then, based on some neighborhood function, a greedy selection is applied between the previous and the new food source. If a better food source is found within this neighborhood, then this becomes the current food source associated with this employed bee.

*3.1.3. Onlooker Bees Phase.* During this phase, employed bees advertise their food sources to the onlooker bees by the means of the waggle dance. Onlooker bees will observe the dance of employed bees and probabilistically choose one of them to follow until their food source. Karaboga et al. [26] suggest the roulette wheel as a way to accomplish this selection, using the quality of the food source as the fitness for this process. After defining which employed bee to follow, the onlooker bee will look for good food sources in the neighborhood of the food source of the employed bee. The same operation for the employed bees phase is used here.

*3.1.4. Scout Bees Phase.* During this phase, an employed bee will become a scout bee if its food source is not improved during a determined number of iterations. Then, the employed bee will abandon its food source and look for a new one.

*3.2. Bee Colony Optimization.* The Bee Colony Optimization (BCO) algorithm is a model inspired by the foraging behavior of honeybees. It is based on the work proposed by Nakrani and Tovey [35] to solve a server allocation problem in datacenters. Two papers were the first to propose similar modifications to the original algorithm to solve a vehicle routing problem: Wong et al. [36] to solve the TSP and Lu and Zhou [37] to solve the TSP.

Lu and Zhou [37] named their algorithm as Bee Collecting Pollen Algorithm, while in Wong et al. [36] they termed their algorithm as BCO. Besides having different names, the main concepts are very similar and the base work is the same. Here we use the proposal of Wong et al. [36] to provide an overview of the algorithm. The main steps of the BCO algorithm are summarized in Box 2 and detailed in the sequence.

```
Initialization
while stopping criterion is not met
  Observe Waggle Dance
  Perform Foraging
  Perform Waggle Dance
End
```

Box 2: Structure of the BCO algorithm and its main steps.

*3.2.1. Initialization.* Each bee represents a candidate solution and is created by using a nearest neighbor or a random approach.

*3.2.2. Observe Waggle Dance.* Before leaving the hive for the foraging process, a bee will decide if it will observe the waggle dance of other bees. A bee is more likely to observe the dance of other bees if its own solution is not as good as the average solution of the hive. By observing the waggle dance, a bee will select another one to follow. The solution of this selected bee will become its preferred solution, to be used during the foraging process. During the first iteration, as bees have no dance to follow, they are allowed to explore the search space without a *preferred solution*.

*3.2.3. Perform Foraging.* During this process, a bee will iteratively construct a full solution to the TSP by moving from one city to another. A bee selects the next city to visit based on a probabilistic transition rule that takes into account the distance between the two cities and if this arc is present in its preferred solution.

*3.2.4. Performing Waggle Dance.* The bee that finds a solution better than its previous one will advertise the new route by means of a waggle dance. The dance has a duration defined by the quality of the solution found by the bee and average quality of the hive.

*3.3. Bee System.* The Bee System (BS) algorithm was initially proposed by Lučić and Teodorović [38] to solve transportation problems. This algorithm is inspired by the honeybees foraging process. The main phases of the BS algorithm are presented in Box 3, as proposed in Lučić and Teodorović [39] to solve the TSP. The original BS algorithm evolved [40] and, together with some structural changes, the authors proposed a new name as the Bee Colony Optimization (BCO). For the present review, the new algorithm is kept under the BS structure and the modifications are discussed in Section 4.

*3.3.1. Changing the Hive Location.* At each iteration, the hive location is changed by placing it on a different city. The hive will be the initial city for each bee. The main idea behind this is that the initial city of the tour being constructed will be different at each iteration.

*3.3.2. Stages.* A stage is considered to be a discrete time unit in the bee's environment. Each iteration has a predetermined number of stages on which the bees will perform some of their tasks.

```
while stopping criterion is not met
  change the hive location
  for each stage
    perform the foraging process
    abandon the food source
    waggle dance
  end for
  local-search
end while
```

Box 3: Structure of the BS algorithm and its main steps.

*3.3.3. Foraging Process.* At each stage, a bee is allowed to visit a predetermined number of cities during the foraging process. This means that, at each stage, it iteratively constructs subtours until reaching a complete solution for the TSP. Not all bees start to forage at the same stage. Those that do not start at the initial stage will follow other bees when they decide to leave the hive. Bees construct subtours by visiting one city after another. The next city to be visited is chosen based on a probabilistic function with the following rules: the greater the distance between the cities, the lower the probability; the greater the number of iterations already spent, the higher the influence of the distance; the greater the number of bees that visited the same link in the past, the higher the probability.

*3.3.4. Abandoning the Food Source.* A bee will decide if it should continue to forage from the same location in the next stage or if it will abandon its subtour. This is a probabilistic function based on the quality of the subtour: the greater the distance, compared with the solutions found by other bees, the higher the probability of abandoning the current solution.

*3.3.5. Waggle Dance.* Bees that have chosen to keep their food sources will now decide if they will recruit other foragers by means of the waggle dance. The decision to dance to advertise its solution is a probabilistic function with low possibility of not dancing. If a bee decided to abandon its solution, it will start to follow another forager bee. It will observe the waggle dance from other bees to decide which one to follow. This decision is guided by a probabilistic function: the lower the cost of the advertised subtour, compared with other solutions, the higher the probability of using it; the higher the number of bees advertising the same subtour, the higher the probability.

*3.3.6. Local Search.* At the end of each iteration, the solution of each bee undergoes a local search. 2opt or 3opt is used, based on the size of the TSP instance.

*3.4. Marriage in Honeybees Optimization.* The Marriage in Honeybees Optimization (MHBO) algorithm was introduced in 2001 by Abbass [41]. In this work, the author

```
Initialization
while stop criteria is not met
  Perform mating fight
  Generate broods
  Improve broods
  Determine the new set of queens
End
```

Box 4: Structure of the MHBO algorithm and its main steps.

described a general-purpose algorithm inspired by the marriage behavior of honeybees, which was then adjusted and tested to solve the SAT problem.

The first works to propose modifications on the MHBO to solve a vehicle routing problem were Yang et al. [42, 43], solving the TSP, and Marinakis et al. [44], solving the CVRP. These works are described in the following section. Here we describe, at a high level, the main steps of the original general-purpose algorithm [41]. The algorithm has four types of agents (bees), each with a different task during the optimization process:

(i) *Queens*: representing the best solutions found so far and used for crossover

(ii) *Drones*: solutions across the search space that are used for crossover

(iii) *Broods*: resulting solutions from crossover and mutation operations between queens and drones

(iv) *Workers*: set of heuristics used to improve broods.

The main steps of the algorithm are highlighted below and the whole process is then summarized in Box 4.

*3.4.1. Initialization.* During the initial phase, a set of queens is randomly generated and a random worker is applied to each of them.

*3.4.2. Mating Flight.* The queen starts the mating flight with an initial energy and speed. At each position in the flight, the queen encounters a different drone, which represents a randomly generated solution. The queen decides if it will mate with the drone based on a probabilistic function as follows: the higher the drone fitness, the higher the probability; the higher the speed, the higher the probability. If the drone is selected for mating, its sperm is added to the queen's spermatheca. A drone sperm represents only part of its solution. During the mating flight, the queen's energy and speed decrease and the mating flight is over when the queen's energy is below a given threshold. The mating flight occurs for every queen in the hive.

*3.4.3. Generating Broods.* Broods are generated by applying crossover and mutation operators between the queens and sperms in their spermathecal.

*3.4.4. Improving the Broods.* During this phase, a worker is used to improve each brood. A fitness is assigned to each worker, which guides their selection: the worker that makes the higher improvement in the broods is more likely to be selected. This fitness is updated throughout the search process based on the quality of the improvement performed on each brood.

*3.4.5. Determining the New Set of Queens.* If the best brood is better than the worst queen, then this brood replaces this queen. This occurs until the best brood is no better than the worst queen.

## 4. Bee-Inspired Algorithms to Solve Vehicle Routing Problems: A Survey

Works relating bee-inspired algorithms with vehicle routing problems are described in this section. They are segmented depending upon the base algorithm, as described previously, and chronologically ordered.

*4.1. Artificial Bee Colony.* In Banharnsakun et al. [34], the authors proposed the first modification found in the literature for the ABC to solve a vehicle routing problem, targeting the TSP. An array of permutations of integers is used to represent each candidate solution. During the employed and the onlooker bees phases the authors used a recombination operator called Greedy Subtour Crossover and applied the 2opt local search after a new solution is created. Random solutions were created during the scout bees phase. Tests were conducted using instances from the TSPLIB, ranging from 51 to 318 cities. Results were compared with other works from the literature and the proposed method achieved better solutions.

Karaboga and Gorkemli [45] proposed an ABC algorithm to solve the TSP. The initial solutions were created using the nearest neighbor approach. During the employed and the onlooker bees phase the authors proposed the usage of a method initially proposed as a mutation operator for a genetic algorithm. The algorithm was tested in two instances from the TSPLIB, with 150 and 200 cities. The results were compared with the ones from a genetic algorithm. Although the proposed algorithm was not able to find the best-known solution in any run, it outperformed the genetic algorithm.

In Singh et al. [46], the authors proposed a hybrid ABC and a genetic algorithm to solve the TSP. Candidate solutions are represented using an array of real numbers and operators are chosen based on this representation. Recombination and mutation phases are added before the employed bees phase. The algorithm was tested using 2 randomly created instances with 30 and 60 cities, respectively. The proposed algorithm was able to produce better solutions than an authors' implementation of the genetic algorithm.

Zhang et al. [47] combined the ABC algorithm with the path relinking method to solve the TSP. During the employed and onlooker bees phase, 2opt is used for any new created solution and path relinking is used whenever a better solution is found. The algorithm was tested in 11 instances from the

TSPLIB, ranging from 51 to 127 cities. Results are compared with those published in other works, specially related to algorithms inspired in the self-organizing maps, and the proposed algorithm outperforms the other ones in regard to the quality of the solution.

In Szeto et al. [48], the capacitated vehicle routing problem (CVRP) was resolved by using the ABC. Candidate solutions are represented with an array of integers, with the depot appearing multiple times. Infeasible solutions are allowed during the search process and a penalty is added to the quality of the solution when a constraint is violated. Initial solutions are created by assigning customers to vehicles by choosing the one that would impose the least cost to the overall solution. During the employed and onlooker bees phase, neighboring solutions are created by using one out of 7 possible operators. Solutions found by the onlooker will not replace its employed bee but the one which is not improved for a long time. Solutions related to scout bees are not randomly generated but are neighbor of the previous solution. Tests were conducted using 34 benchmark instances from the TSPLIB, ranging from 50 to 483 customers. The obtained results are good, with the algorithm being able to find the best-known solutions on many runs for different instances.

The Green TSP (GTSP) was resolved in Özceylan et al. [49] with an algorithm hybrid between the Ant Colony Optimization (ACO) and the ABC. ABC is used to determine the best speed between two visited customers. Tests with the hybrid algorithm were performed with 36 instances proposed by the author, ranging from 5 to 40 cities. Results were compared with those proposed by the software LINGO. The proposed algorithm provides better solutions.

Ji and Wu [50] tested the ABC algorithm with different mutation operators as the neighboring operator for the employed and onlooker bees phases: random swap; random insertion; random swap of subsequence; random insertion of subsequence; random reversing of subsequence; random reversing swap of subsequence; and random reversing insertion of subsequence. 2opt and 3opt local search heuristics are used at the end of the process to improve the final solution. Based on the tests made, the best results were obtained with the insertion operators.

In Li et al. [51], the authors proposed a hybrid algorithm combining the ABC algorithm with a nearest neighbor method to solve the TSP. To assess the performance of their proposal they applied the algorithm to a number of instances from the TSPLIB and compared it with the BCO algorithm, showing a superior performance for the instances evaluated.

In Brajevic [52], the author proposed some modifications in the ABC algorithm so that it could be applied to the capacitated vehicle routing problem. The modifications include the proposal of a specific encoding scheme, constraint handling, and specific neighborhood operators. The method was applied to 12 small-scale benchmark CVRP instances and the results were compared to those of the best known solutions.

Shi et al. [53] proposed the ABC-T algorithm for solving the VRPTW. ABC-T is based on the standard ABC but uses a tournament selection mechanism to select food sources, improving the global search capability of the algorithm. Their proposal was assessed using the Solomon's R102 problem instance.

Iqbal and Rahman [54] tackled the Vehicle Routing Problem with Time Windows (VRPTW) using the ABC algorithm. Candidate solutions are represented with an array of integer numbers, where the depot appears as much times as vehicles are used in the solution. Infeasible solutions are allowed during the search, but this implies penalties to the quality of the solution. During the employed and onlooker bees phase, neighboring solutions are created by swap mutation. Also, when a better solution is found during these phases, it does not replace the original solution but the worst one in the population that has not been improved for some time. Onlooker bees create new solutions by using the swap operator on the solution to be discarded. Tests were conducted on 28 randomly generated instances proposed by the authors, ranging from 20 and 300 customers. The modified algorithm was compared with the original one and the new implementation was able to provide better solutions.

In Karabulut and Tasgetiren [55], a modified ABC algorithm was proposed to solve the TSP with Time Windows (TSPTW). Initial solutions were randomly constructed. In the employed and onlooker bees phase, neighboring solutions were chosen using a destruction and construction method and then undergoing a local search. For the onlooker bee phase, a new solution was constructed using the same method as the employed bees. The proposed algorithm was tested in 55 benchmark instances from the literature, ranging from 20 to 200 cities. The results were compared with other two works from the literature, presenting competitive solutions.

Bhagade and Puranik [56] proposed an ABC algorithm to solve the TSP. There is no description about the necessary modifications in the original algorithm. The proposed algorithm was tested in 4 instances created by the authors. The results were compared with another unmentioned method, which was outperformed by the proposed algorithm.

Pathak and Tiwari [57] tackled the TSP by using the original version of the ABC. Since candidate solutions are represented using arrays of real valued numbers, a map function is required to translate this array into a TSP solution. For that purpose they use a technique called shortest position value. Tests were conducted using 2 instances proposed by the authors, with 30 and 60 cities, respectively. Results obtained by the proposed algorithm were better than an authors' implementation of a genetic algorithm.

In Li et al. [58], the authors proposed the use of a swap operator in the ABC algorithm to help the bees in finding better candidate tours for the TSP based on a greedy search. To assess the performance of their algorithm they used 6 TSP instances from the literature and compared the results with the PSO algorithm. They also performed a sensitivity analysis of some ABC algorithm parameters.

In Bin et al. [59], the Vehicle Routing Problem with Soft Time Windows (VRPSTW) was resolved using the ABC algorithm. A candidate solution is represented using a set of arrays with integer values, on which each array represents the route of a vehicle. Initial solutions are randomly created. During the employed and onlooker bees phases,

neighborhood solutions are created by performing inter- and intraroute modifications. Tests were conducted on 56 benchmark instances from the literature. The results are compared with other 5 algorithms from the literature. For most of the tests, the proposed algorithm outperforms the other ones.

Kiran et al. [60] modified the structure of ABC algorithm adding a crossover step between the employed and the onlooker bees phase. The modification is mainly described to tackle continuous optimization problem and later they suggest the same structure to deal with the TSP. Although tests were made and results were posted to the TSP, the specific modifications to deal with this problem are not clear. Tests were conducted with 3 instances proposed by the authors, ranging from 10 to 30 cities, and the proposed algorithm was able to find solutions better than the original algorithm.

Sabet et al. [61] proposed an implementation of the ABC to solve the TSP. Initial solutions are built by using a probabilistic implementation of the nearest neighbor. Mutation operators are used to create neighboring solutions during the employed and onlooker bees phases. Tests were conducted using 5 instances of the TSPLIB, ranging from 51 to 99 cities. The algorithm was capable of finding good results but did not outperform the results from other algorithms from the literature chosen by the authors.

In Yao et al. [62] the authors proposed modifications on the ABC algorithm in order to solve the Periodic Vehicle Routing Problem (PVRP). Recombination and mutation operators are used in order to define new solutions during the search process. 2opt is used to improve the routes found. Tests were made on 9 benchmark instances from the literature with the following characteristics: 50 to 100 customers; 1 to 6 vehicles; and periodic cycles from 2 to 10 days. Results were compared with those published in other works and the algorithm has been shown to be competitive in regard to the quality of the solutions found.

Liu [63] proposed a merge between an Adaptive GA (AGA) and the ABC algorithm to solve the Multidepot Vehicle Routing Problem with Time Window (MDVRPTW). In the description of the algorithm, it is not clear how the algorithm is improved with the ABC concepts. Tests are conducted with 20 benchmark instances from the literature and results are good.

In Singh et al. [46], a solution based on the ABC algorithm is proposed to the 2-dimensional Loading Capacitated Vehicle Routing Problem (2L-CVRP). Candidate solutions are represented using an array of real numbers, which requires a decode process to validate the quality and feasibility. 2opt and 3opt local search algorithms are used to improve the routes. A set of 3 different heuristics are used to deal with the 2-dimensional loading problem. Tests were conducted using 50 benchmark instances from the literature with the following characteristics: 15 to 100 customers; 15 to 310 demanded goods. In most of the tests the proposed algorithm outperformed others from the literature.

Yang and Pei [64] combined the ABC algorithm with the PSO to solve the travelling salesman problem but used PSO as the main algorithm in the problem solution due to its claimed faster convergence.

Pandey and Kumar [65] introduced some enhancements by adding crossover operators to the ABC algorithm and applied it to the TSP, evaluating its performance in terms of efficiency and accuracy. Their hybrid algorithm works in five phases: initialization; employed bee phase; crossover; onlooker bee phase; and scout bee phase. They applied their algorithm for different values of the maximum cycle number and different dimensions of the individuals in the population.

Rekaby et al. [66] proposed an Adaptive Artificial Bee Colony (AABC) algorithm applied to the travelling salesman problem.

In Yuan and Zhu [67], the authors proposed a hybrid ABC with a genetic algorithm to solve the TSP. It is not clear how each phase of the ABC algorithm was modified to tackle the TSP. The GA inspiration was used as an additional step by adding crossover and mutation operators after the scout bees phase. The algorithm was tested using an instance with 30 cities from the TSPLIB. The result was compared with other 5 works from the literature and the proposed algorithm outperformed the other ones.

In Zhang et al. [68], the Environmental Vehicle Routing Problem (EVRP) is resolved using a hybrid algorithm between ABC and genetic algorithms. Employed and onlooker bees phases are modified by adding crossover and mutation operators. Also, 2opt and 3opt are used to improve the routes created. Tests were conducted with 19 benchmark instances from the literature, ranging from 31 to 51 customers and 3 to 7 vehicles.

In Nahum et al. [69], the authors proposed modifications in the ABC to solve the Multiobjective VRPTW (MOVRPTW). The authors proposed a Vector Evaluated ABC (VEABC) so that multiobjective problems can be properly solved. The vector evaluated approach is borrowed from genetic algorithms, where portions of the population are evaluated on each objective. Candidate solutions are represented as an array of integers, with the depot appearing multiple times. During the employed and onlooker bees phase, neighboring solutions are created by using an operator chosen from a pool of 9 different ones. Tests were conducted on 55 benchmark instances from the literature. The algorithm was capable of finding good solutions with regard to the cost of routes and number of vehicles.

Kocer and Akca [70] used a local search method to improve the performance of the ABC algorithm for solving the TSP. They used a loyalty function of BCO as a fitness function for the ABC algorithm, applied their proposal to TSP instances from the TSPLIB, and compared their results with those of other bee-inspired algorithms.

In Chung [71], the authors combined the ABC algorithm with a sweep method to quickly generate near optimal solutions to the capacitated vehicle routing problems. His proposal was compared with a genetic algorithm over 60 instances containing 100 nodes.

In the proposal of Nagaya and Inoie [72], the authors created a hybrid between the ABC and the simulated annealing (SA) algorithm to solve capacitated vehicle routing problems. They assessed their algorithm in 11 benchmark instances and

compared the results with that of the standard ABC, SA, and the best-known solution.

Zhang and Lee [73] introduced a routing directed ABC (RABC) algorithm for solving the capacitated vehicle routing problem (CVRP) and tested their proposal in various benchmark problems from the literature.

Gündüz et al. [74] proposed a hierarchical approach based on the Ant Colony Optimization (ACO) and the ABC algorithms to solve the TSP. In their approach the ACO is used as a path-construction method and the ABC as a path-improvement method. As such, the ACO provides a better initial solution for the ABC algorithm, which becomes responsible for seeking the optimal solution. The authors used 10 TSP instances from the literature and showed that their hybrid proposal achieves better quality solutions than the individual use of ACO or ABC, with less computational time.

Alzaqebah et al. [75] used a modified ABC algorithm for solving Vehicle Routing Problems with Time Windows (VRPTW). Their modification included the use of abandoned solutions by scout bees, a roulette wheel selection of candidate solutions, and the generation of new random candidate solutions to be inserted in the swarm. The proposal was applied to the Solomon benchmark datasets and the results were compared with that of the standard ABC.

*4.2. Bee Colony Optimization.* One of the first proposals using the BCO algorithm to resolve a vehicle routing problem, the TSP, is described in Lu and Zhou [37]. There are only a few specificities from this work to the algorithm described in the previous section: during the iterations, the starting position of the bees changes randomly; new bees are introduced in the population during the search process. Tests are made using two instances from the TSPLIB, with 51 and 70 cities, respectively.

Wong et al. [36] is one of the first works to relate the BCO to a vehicle routing problem, the TSP. This is the work used in the previous section to describe the BCO algorithm. Tests were conducted using 15 instances from the TSPLIB, ranging from 48 to 318 cities. Results were compared with other 6 works from the literature which was unable to provide better results.

In Wong et al. [76–79], the authors tackled the TSP by proposing some modifications on Wong et al. [36] to improve the efficiency of the algorithm. The modification resides on the local search used after the foraging phase. A method named frequency-based pruning strategy (FBPS) chooses which solutions should undergo a local search, which is an efficient implementation of the 2opt, called fixed-radius near neighbor (FRNN) 2-opt. Tests were made using 84 instances from the TSPLIB, ranging from 14 to 1379 cities. The results are good, decreasing the computational time without compromising the quality of the solution. The authors make an extensive comparison of the quality of the solutions with other works from the literature and the proposed algorithm is capable of finding competitive solutions. A generic framework for combinatorial optimization problems, including vehicle routing problems, is proposed in Wong et al. [78, 79].

In Wong et al. [77], the authors expanded the work in Wong et al. [76] to solve the TSP, by changing the transition rule in order to make it quicker, in terms of computational time. Instead of visiting one city at a time, the bee is now allowed to visit a set of cities (a subroute). The proposed algorithm is tested on 84 instances from the TSPLIB, ranging from 14 to 1379 cities. The modification decreased the running time without compromising the quality of the solution.

In Häckel and Dippold [80] the authors proposed a two-stage BCO-like algorithm to tackle the Vehicle Routing Problem with Time Windows (VRPTW). During the first stage it determined the number of vehicles and the cities to be visited. The route of each vehicle was created during the second stage. The proposal was tested using 56 benchmark instances from the literature, with 100 customers each. The results were compared with other works from the literature and the proposed algorithm was not able to find competitive solutions, in comparison to other algorithms.

In Özceylan et al. [49], the authors proposed a modification on the BCO to solve the TSP. 3opt local search is added after the foraging phase in order to improve the routes built. Tests were conducted with 16 instances proposed by the authors and the results were compared with those from the nearest neighbor to represent the improvement made by the proposed algorithm.

In Girsang et al. [81], the authors proposed modifications in the BCO to solve the TSP. The modifications were to enhance the computational performance by using a method called pattern reduction. Two pattern reduction operators were added and the modified algorithm was compared with the original BCO in 7 instances from the TSPLIB, ranging from 51 to 1002 cities. The modifications significantly reduced the computational time with a small reduction in the average quality of the solutions.

*4.3. Bee System.* In Lučić and Teodorović [39] and in Lučić and Teodorović [82], the authors proposed the Bee System algorithm to solve the TSP. These are the works referred in the previous section to describe the main steps of this algorithm. In Lučić and Teodorović [39], tests were conducted using 8 instances from the TSPLIB, ranging from 51 to 280 cities. Results were good, with the algorithm being able to find the best-known solution for 6 instances, with size lower than 102 cities. In Lučić and Teodorović [82], the tests were conducted on 10 instances, ranging from 51 to 1002 cities.

In Lučić and Teodorović [83], the authors expanded the work in Lučić and Teodorović [39] so that the BS is applicable to the Stochastic Vehicle Routing Problem (SVRP). The proposal uses the original BS to create a "giant tour," such as a solution to the TSP. Based on fuzzy rules, the giant tour is then divided into subtours, one for each vehicle. TSP instances from the TSPLIB were used to create instances for the stochastic VRP. The authors used 10 instances, ranging from 51 to 1002 customers. The algorithm was able to find good average solutions.

In Teodorovic et al. [40], the Bee System is described as a variation of a concept the authors call Bee Colony Optimization (BCO). The BS is slightly modified so that all the operations are conducted within two major steps: (1)

forward pass and (2) backward pass. During the forward pass bees will construct partial solutions and in the backward pass bees will communicate with each other and decide if they will continue with their own solutions or become followers. The algorithm was tested on 6 TSP instances ranging from 51 to 280 cities.

Nikolić et al. [84] follows the concept described in Teodorovic et al. [40] and proposes the use of the BCO to solve the VRPTW. In this implementation, the structure of the algorithm has evolved from a constructive to an improvement algorithm. In the current concept, each bee is associated with a solution which is iteratively modified targeting the improvement of the solution. The structure of the forward and backward passes is maintained. However, in the forward pass, instead of building a solution, modification operators are used targeting the improvement of the current solution. The algorithm was tested on Solomon's benchmark instances and the algorithm was capable of finding satisfactory solutions. The authors suggest the results could be improved by using local search techniques together with the proposed algorithm.

In Jawarneh and Abdullah [85], the authors proposed an improvement in the algorithm introduced in Nikolić et al. [84] to solve the VRPTW. The two main improvements are the use of a sequential insertion heuristic to create the initial population and the online tuning of parameter's values. The proposed algorithm was tested on Solomon's benchmark instances. A direct comparison made by the authors with the original algorithm showed that the proposed method is capable of finding better results.

*4.4. Marriage in Honeybees Optimization.* In Yang et al. [42], the authors proposed an MHBO algorithm to solve the TSP. The main addition highlighted by the authors is the usage of the Wolf Pack Search (WPS) heuristic to improve the queen and the drone solutions. The algorithm was tested on two instances from the TSPLIB, with 16 and 48 cities, respectively. The results were compared with the original algorithm without the WPS and an authors' implementation of a genetic algorithm. The proposed algorithm was able to find better solutions.

In Yang et al. [43], the authors tackled the TSP with an implementation of the MHBO. It uses the Nelder-Mead method as the heuristic for the local search phase. Tests were performed on 2 instances from the TSPLIB, with 16 and 48 cities, respectively.

In Marinakis et al. [86], the authors used the proposal in Marinakis et al. [44] to solve the TSP. The initial population is created using the Multiple Phase Neighborhood Search-GRASP (MPNS-GRASP). To generate new broods, a crossover inspired on the adaptive memory procedure is proposed, which selectively copies information from the queen and the drones. As the workers, they use an approach called Expanding Neighborhood Search, which applies a set of local search strategies to the target solution. The algorithm was tested on 74 instances from the TSPLIB, ranging from 51 to 85900 cities. From 51 to 783 cities, the algorithm has found the best solutions on all runs. For the other instances, the algorithm was capable of finding either the best solution or a very close one on all runs.

In Celik and Ulker [87], the authors applied the MHBO to solve the Asymmetric Travelling Salesman Problem (ATSP), that is, the one in which the cost from A to B is different from the cost from B to A. The algorithm was applied to three different instances of the ATSP and the results were compared with that of a genetic algorithm and a simulated annealing method. They showed that the MHBO algorithm was capable of finding the best-known solution in most of the runs.

A solution using the MHBO algorithm to the capacitated vehicle routing problem with three-dimensional loading constraints (3L-CVRP) is proposed in Ruan et al. [88]. The initial population is built by using Multiple Phase Neighborhood Search Greedy Randomized Adaptive Search Procedure (MPNS-GRASP) with the best solution being chosen as the queen. The mating of queen and drones is made using the 2-point crossover. As workers, local search algorithms are used to make inter- and intraroute changes to improve the overall solution. To tackle the loading part of the problem, 6 different loading heuristics are used to determine if packages can be loaded on vehicles. Tests were conducted on 27 instances from the literature, ranging from 16 to 101 customers. Results were compared with other 3 works from the literature and, in most cases, the proposed algorithm outperforms the other ones.

Table 2 presents a chronological summary of the works reviewed, organized in terms of the base algorithm and target problem.

## 5. TSPoptBees: A Bee-Inspired Algorithm for Solving the Travelling Salesman Problem

The optBees algorithm was originally proposed by Maia et al. [21, 90] to solve continuous optimization problems inspired by the collective behavior of honeybees during the foraging process. It was later applied to other optimization problems, such as data clustering [91] and the training of artificial neural networks [92].

The TSPoptBees is a modification of the original optBees algorithm to solve the travelling salesman problem. It was originally introduced in a short conference paper by the same authors [93] and we now bring a much more complete explanation of the algorithm, plus a new set of experiments to validate the proposal.

In the TSPoptBees, each bee represents a route to the TSP, which is a candidate solution to the target problem. The quality of the food source explored by a bee is related to the quality of the associated TSP route in such a way that the higher the quality, the lower the total travelling cost. As in the original algorithm, there are three types of bees: (1) *recruiters*, which are responsible for recruiting other bees to exploit promising regions of the search space; (2) *scouts*, that are responsible for exploring new regions of the search space; and (3) *recruited*, which follow the recruiter bees to exploit new solutions in their surroundings.

The bees fly around the space of candidate solutions searching for the high quality ones. According to the quality

Table 2: Reviewed works using bee-inspired algorithms to solve vehicle routing problems.

| Base algorithm | Target problem | Year | Authors |
|---|---|---|---|
| Artificial Bee Colony (ABC) | TSP | 2010 | Banharnsakun et al. [34] |
| | | 2011 | Karaboga and Gorkemli [45] |
| | | | Singh et al. [46] |
| | | | Zhang et al. [47] |
| | | | Li et al. [51] |
| | | 2012 | Bhagade and Puranik [56] |
| | | | Pathak and Tiwari [57] |
| | | | Li et al. [58] |
| | | 2013 | Kiran et al. [60] |
| | | | Verma and Kumar [25] |
| | | | Sabet et al. [61] |
| | | | Pandey and Kumar [65] |
| | | | Rekaby et al. [66] |
| | | | Yang and Pei [64] |
| | | 2014 | Yuan and Zhu [67] |
| | | | Kocer and Akca [70] |
| | | 2015 | Gündüz et al. [74] |
| | GTSP | 2011 | Özceylan et al. [49] |
| | TSPTW | 2012 | Karabulut and Tasgetiren [55] |
| | CVRP | 2011 | Szeto et al. [48] |
| | | | Brajevic [52] |
| | | 2014 | Chung [71] |
| | | | Nagaya and Inoie [72] |
| | | 2015 | Zhang and Lee [73] |
| | EVRP | 2014 | Zhang et al. [68] |
| | PVRP | 2013 | Yao et al. [62] |
| | TDVRP | 2011 | Ji and Wu [50] |
| | VRPTW | 2012 | Shi et al. [53] |
| | | 2016 | Alzaqebah et al. [75] |
| | VRPSTW | 2012 | Iqbal and Rahma [54] |
| | MOVRPTW | 2014 | Nahum et al. [69] |
| | MDVRPTW | 2013 | Liu [63] |
| | 2L-CVRP | 2013 | Bin et al. [59] |
| Bee Colony Optimization (BCO) | TSP | 2008 | Lu and Zhou [37] |
| | | | Wong et al. [36] |
| | | 2009 | Wong et al. [76, 77] |
| | | 2010 | Wong et al. [78, 79] |
| | | 2012 | Singh and Narayan [89] |
| | | | Girsang et al. [81] |
| | VRPTW | 2009 | Häckel and Dippold [80] |
| Bee System (BS) | TSP | 2002 | Lučić and Teodorović [39] |
| | | 2003 | Lučić and Teodorović [82] |
| | | 2006 | Teodorovic et al. [40] |
| | SVRP | 2003 | Lučić and Teodorović [83] |
| | VRPTW | 2013 | Nikolić et al. [84] |
| | | 2015 | Jawarneh and Abdullah [85] |
| Marriage in Honeybees Optimization (MHBO) | TSP | 2007 | Yang et al. [42] |
| | | | Yang et al. [43] |
| | | 2012 | Celik and Ulker [87] |
| | VRP | 2011 | Marinakis et al. [86] |
| | | 2008 | Marinakis et al. [44] |
| | | 2013 | Ruan et al. [88] |

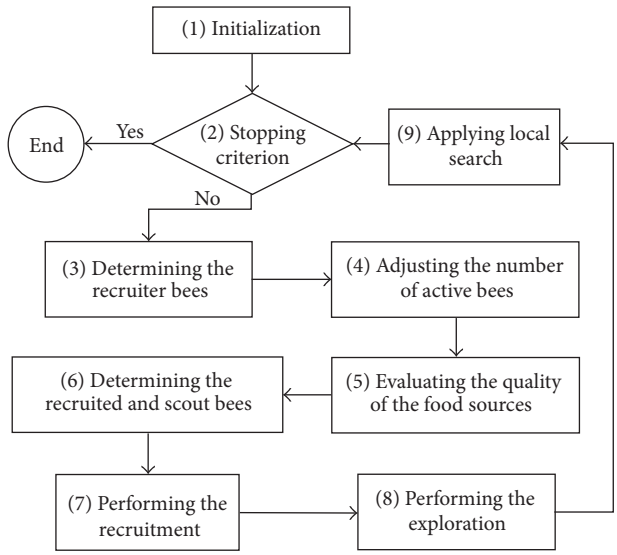



Figure 1: Flowchart of the TSPoptBees algorithm.

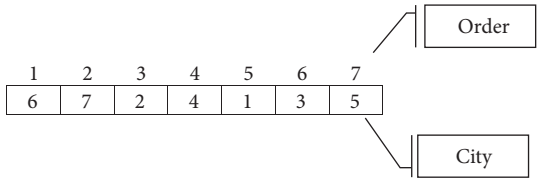



Figure 2: Solution representation for the TSP.

of the current solutions, multiple bees may be classified as recruiters. Some of the other bees are recruited by the recruiter bees to exploit their corresponding food sources, while the other ones become scout bees and explore new regions of the search space. If the number of promising regions is high, new bees can be activated to participate in the recruitment process. A high level flowchart of the TSPoptBees is presented in Figure 1 and each of the steps is then described.

*5.1. Representation of a Candidate Solution.* Each bee represents a candidate solution for the TSP, which is represented by the permutation of $n$ integers. The example in Figure 2 represents a candidate solution that passes by the cities in the following order: 6, 7, 2, 4, 1, 3, 5, and returning to 6.

*5.2. Initialization.* In this step, the initial population of bees is built by using a set of construction heuristics [29], as follows:

(i) One solution is created using a Greedy algorithm.

(ii) One solution is created using the Cheapest Insertion algorithm.

(iii) Multiple solutions are created using the nearest neighbor algorithm. Each solution is created using a different initial city, meaning that it can create up to $n$ different solutions.

(iv) If necessary, additional solutions are created by taking random routes created by the above algorithms after

applying one of the modification operators described in the Exploration Step.

*5.3. Stopping Criterion.* The stopping criterion terminates the search process when it completes a certain number of iterations without improving the best solution; that is, it is based on the stabilization of the search process.

*5.4. Determining the Recruiter Bees.* In this step there is a probabilistic selection of the recruiter bees based on the quality of the food source being explored by each bee. The main idea is that the better the quality of the food source, the greater the probability of the bee exploring this food source to become a recruiter.

The probability $p_i$ of bee $i$, associated with a solution cost $\mathrm{sc}_i$, becoming a recruiter is determined by

$$p_i = \left( \frac{(1 - p_{\min})}{(\mathrm{sc}_{\max} - \mathrm{sc}_{\min})} \right) * (\mathrm{sc}_{\max} - \mathrm{sc}_i) + p_{\min}. \quad (5)$$

In (5), $p_{\min}$ is the minimum probability of a bee to become a recruiter and $\mathrm{sc}_{\max}$ and $\mathrm{sc}_{\min}$ represent, respectively, the maximum and minimum solution cost among all bees. Then, the probability $p_i$ is compared to a random number $n_{\mathrm{random}}$ within the interval [0, 1] in such a way that if $n_{\mathrm{random}} < p_i$, then bee $i$ becomes a recruiter. To avoid multiple recruiter bees acting around the same promising region, an *inhibition radius* is used. Then, each recruiter is considered, in ascending order with regard to the solution cost $\mathrm{sc}_i$, and the remaining recruiters that are within the inhibition radius $\rho$ are labeled as nonrecruiters. In other words, given the distance $d_{i,j}$ between bees $i$ and $j$, if $d_{i,j} < \rho$, then bee $j$ is not a recruiter, assuming that bee $i$ has a higher fitness than bee $j$.

The distance between two bees is calculated by a function based on their solution cost of the two bees being compared as in

$$d_{i,j} = \frac{\mathrm{abs}\left(\mathrm{sc}_i - \mathrm{sc}_j\right)}{\max\left(\left[\mathrm{sc}_i; \mathrm{sc}_j\right]\right)}. \quad (6)$$

In (6), sc is the cost of a specific solution, abs(·) is a function that returns the absolute value of the argument, and max(·) is a function that returns the highest value from a set of arguments.

*5.5. Adjusting the Number of Active Bees.* In this step, the number of active bees needed during the current iteration is determined, increasing or decreasing it based on the variety of food sources being explored. Let $n_r$ be the number of recruiter bees, $n_{\mathrm{act}}$ the number of active bees in the current iteration, and $n_m$ the expected number of nonrecruiters (recruited and scout) for each recruiter. Hence, $n_d = (n_r + 1) * n_m$ is the expected number of active bees in the current iteration. If $n_d > n_{\mathrm{act}}$, then $n_{ad} = n_d - n_{\mathrm{act}}$ is the number of bees to become active. If $n_d < n_{\mathrm{act}}$, then $n_{ad} = n_{\mathrm{act}} - n_d$ is the number of bees to be deactivated; $n_d \in [n_{\min}, n_{\max}]$. When necessary, new bees are included in the population (activated) by randomly selecting $n_{ad}$ bees already in the population and applying one of the modification operators described in the Exploration Step. When bees are deactivated, the $n_{ad}$ bees with the highest cost (worst solutions) are removed from the population.

*5.6. Evaluating the Quality of the Food Sources.* The algorithm aims to maximize the quality of the food source explored by each bee. In this step, a function that calculates the cost of all solutions and transforms them into a quality factor is used, as in (7), meaning that the lower the cost, the higher the quality.

$$q_i = 1 - \frac{(\mathrm{sc}_i - \mathrm{sc}_{\min})}{(\mathrm{sc}_{\max} - \mathrm{sc}_{\min})}. \quad (7)$$

In (7), $q_i$ is the quality associated with the solution of bee $i$, $\mathrm{sc}_i$ is the cost of the solution associated with bee $i$, and $\mathrm{sc}_{\max}$ and $\mathrm{sc}_{\min}$ are, respectively, the maximum and minimum solution costs among all bees.

*5.7. Determining the Recruited and Scout Bees.* In this step the recruiter and scout bees are determined. Moreover, for each recruited bee, its recruiter is also defined such that the number of recruited bees for each recruiter is proportional to the quality of the food source associated with the recruiter. Consider $n_{\mathrm{nr}} = n_{\mathrm{act}} - n_r$ as the number of nonrecruiter bees. The number of recruited bees in the current iteration is $n_{\mathrm{rtd}} = \mathrm{round}(p_{\mathrm{rec}} * n_{\mathrm{nr}})$, in which round(·) is a function that gives the nearest integer and $p_{\mathrm{rec}}$ is the percentage of nonrecruiter bees that are going to be recruited. Thus, the number of scout bees is $n_{\mathrm{ex}} = n_{\mathrm{nr}} - n_{\mathrm{rtd}}$. Consider $Q_r$ as the sum of the qualities of all recruiter bees; then $\mathrm{nr}_i = \mathrm{round}((q_i/Q_r) * n_r)$ is the number of recruited bees associated with recruiter bee $i$. Nonrecruiter bees are then associated with the recruiters using a roulette wheel approach based on the quality of the nonrecruiter bees, being tagged as recruited bees.

*5.8. Performing the Recruitment.* The recruitment consists of exploiting new food sources in the neighborhood of the recruiter bees by attracting the recruited ones. This is accomplished by combining the solution associated with the recruited bee with the solution associated with its recruiter, creating a new solution around this neighborhood. The combination of two solutions was implemented with a set of *crossover operators*, borrowing ideas from evolutionary algorithms. Three variations of the Order Crossover [94] are used:

(i) Type 1 (see Figure 3(b)): 50% to 80% of the recruited bee is copied as a single fragment to the final solution, while the remaining cities are copied, in order, from the recruiter.

(ii) Type 2 (see Figure 3(c)): 50% to 80% of the recruited bee is copied as a single fragment to the final solution, while the remaining cities are copied, based on the edges, from the recruiter.

(iii) Type 3 (see Figure 3(d)): 50% to 80% of the recruited bee is copied from two separated fragments to the

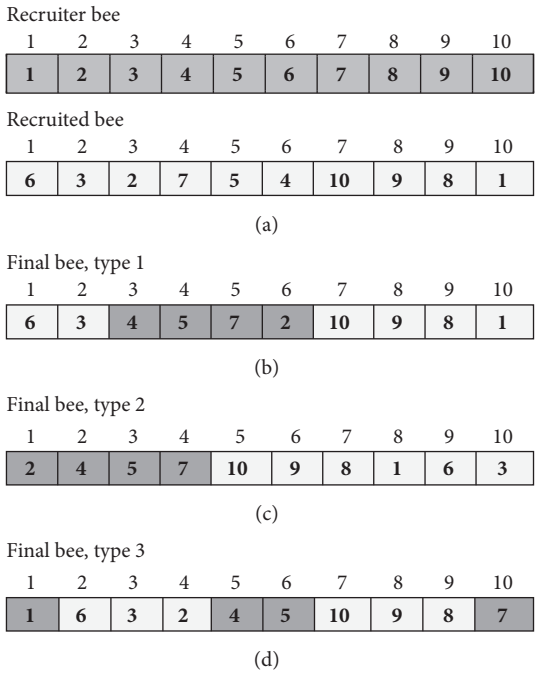



Figure 3: Example of a final solution after the possible crossover-like operators using the same recruiter and recruited bees.

final solution, while the remaining cities are copied, in order, from the recruiter.

For each pair of recruiter and recruited bees, the above operators are randomly chosen with the same probability.

After a new solution is created, it will be used by the recruited bee, replacing its current solution, only if there was an improvement. In addition, if the quality of the new solution associated with the recruited bee is better than the one associated with its recruiter, this solution is assigned to the local search, as will be explained later.

*5.9. Performing the Exploration.* In this step, the scout bees look for new food sources. Exploration operators that provide a new food source were implemented as a set of mutation operators, also borrowing ideas from evolutionary algorithms [95]. The operators are as follows:

(i) Swap of cities: randomly choose two cities and switch their positions.

(ii) City insertions: randomly choose a city and insert it in a new, random position.

(iii) Subroute inversion: randomly choose two cities and invert the subroute between them.

(iv) 3opt movement: perform a 3opt movement by randomly choosing the breaking and joining points.

For each scout bee, a random solution is chosen from the population and goes through one of the above operators, which is chosen randomly with the same probability.

*5.10. Applying Local Search.* In this step, the 2opt local search with the "first improvement" option is used to improve promising solutions [29]. This is limited to a maximum of $2 * n^2$ iterations at each run, in which $n$ is the number of cities of the instance being solved.

Only a few selected recruited bees are assigned to the local search at each iteration, which occurs during the recruitment phase when the recruited bee finds a food source better than the one associated with its recruiter.

## 6. Performance Assessment

To assess the performance of TSPoptBees, several experiments were conducted using benchmark instances from the TSPLIB (available online at http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/) [7]. In total, 28 instances, ranging from 48 to 439 cities, were used and the optimal solution is known for each of them. The proposed algorithm was implemented in MATLAB and executed on an Intel Core i5 1.9 GHz with 8 GB RAM. For each instance, the algorithm was executed 30 times.

Before running the TSPoptBees, the value for a set of parameters must be defined. Further details on how these parameters can be tuned were presented in Masutti and de Castro [96] and the values used in the present experiments are presented below:

(i) $n_{\min} = 50$: minimum number of active bees, which is also the size of the initial population.

(ii) $n_{\max} = 300$: maximum number of active bees.

(iii) $n_m = 18$: average number of nonrecruiter bees (recruited and scout) for each recruiter.

(iv) $p_{\min} = 1\%$: minimum probability of an active bee to become a recruiter.

(v) $p_{\text{rec}} = 70\%$: percentage of recruited bees among the nonrecruiters (recruited and scout).

(vi) $\rho = 0.1$: social inhibition radius.

(vii) maxIt = 1000: maximum number of iterations without an improvement of the best solution.

*6.1. Assessing the Computational Cost Based on the Number of Customers.* The computational cost analysis of TSPoptBees was done empirically when executing the algorithm for two instances of TSPLIB: fl417 and pr439 were divided into another 20 instances, 10 for each original TSPLIB instance, varying in size according to sets [25 50 75 100 150 200 250 300 400 417] and [25 50 75 100 150 200 250 300 400 439], respectively. Using city sets from the same instance for this test presented more reliable results than using different instances with different sizes, since each instance may have particular characteristics that influence the convergence of the search process. Two different instances, but of similar sizes, were used to reduce the bias due to the characteristics of a single instance.

For each instance described above, TSPoptBees was run 30 times and only the characteristics related to the computational effort of the algorithm were used in the

Table 3: TSPoptBees computational cost evaluation for TSP. $n$ is the number of cities in the instance; Iterations is the average number of iterations for convergence; $n_{act}$ is the average number of bees in the final swarm; Time is the average time, in seconds, for running the algorithm.

| $n$ | fl417 | | | pr439 | | |
|---|---|---|---|---|---|---|
| | Iterations | $n_{act}$ | Time | Iterations | $n_{act}$ | Time |
| 25 | 1402.63 | 170.13 | 14.98 | 1402.63 | 207.53 | 19.10 |
| 50 | 1657.60 | 203.87 | 28.50 | 1560.57 | 242.73 | 34.06 |
| 75 | 2034.80 | 253.73 | 55.55 | 2170.07 | 259.60 | 63.78 |
| 100 | 2402.63 | 277.93 | 95.29 | 2234.60 | 264.00 | 83.72 |
| 150 | 3110.63 | 311.67 | 195.43 | 3692.37 | 287.47 | 200.83 |
| 200 | 4182.17 | 369.60 | 381.15 | 4065.37 | 282.33 | 293.47 |
| 250 | 4927.83 | 426.07 | 617.87 | 5299.03 | 300.67 | 474.90 |
| 300 | 5636.97 | 473.73 | 1006.08 | 4742.80 | 309.47 | 576.66 |
| 400 | 6085.43 | 486.20 | 1726.61 | 4821.03 | 321.20 | 961.55 |
| 417 | 5746.30 | 495.00 | 1638.32 | — | — | — |
| 435 | — | — | — | 7591.30 | 289.67 | 1719.78 |

analysis of the results. These characteristics are (1) the average number of iterations for convergence, (2) the mean size of the final swarm, and (3) the average running time. It is worth mentioning that the obtained results are specific to the implementation used, since each operator can present a different computational complexity.

Table 3 presents the results obtained for the two instances used. For each instance, the average results of the number of iterations for convergence of the search process, the number of active bees in the final swarm, and the execution time of the algorithm are presented. According to the results presented, the number of active bees in the final swarm tends to increase along with the growth of the number of cities in the instance. However, growth seems to stabilize from a certain value of $n$. This behavior is observed for the two instances used in this experiment. For the fl417 instance, the growth is more accentuated for smaller values of $n$, but it seems to stabilize around $n = 300$. For the instance pr439, growth exists, however, in a milder fashion. An interesting point is that the value of $n_{act}$ decreased for instance pr439 between $n = 400$ and $n = 439$ (actual instance size).

For the number of iterations for convergence (Iterations in Table 3), the behavior is also similar for the two instances. The number of iterations for $25 \leq n \leq 250$ is very close between the two instances, presenting a marked increase. From this value, there is a different behavior between the two instances. For fl417, there is a smaller growth between $n = 250$ and $n = 400$ and a slight decrease for $n = 417$ (actual instance size). For pr439, there is a decrease in the number of iterations between $n = 250$ and $n = 300$, almost constant between 300 and 400, and an abrupt increase between 400 and 439.

The runtime also has similar characteristics between the two instances and can be seen as a composition of the two attributes described above. For the two instances, the values are very similar between $n = 25$ and $n = 150$, presenting a certain growth in the execution time. From this point, there is a more pronounced growth for the fl417 instance, which follows a behavior similar to that seen for the number of bees. However, the runtime decreases between $n = 400$ and $n = 417$, the same behavior seen for the number of iterations. For the pr439 instance, there is a less pronounced growth, which changes noticeably between $n = 400$ and $n = 439$, with an abrupt increase, as well as the observed number of iterations.

It is evidenced by the results that, in relation to the computational effort, the algorithm can present different values for different instances. However, what can be observed is a behavior trend that is independent of the instance being solved. Figure 4 shows the runtime-related trend for the two instances. It is observed that the running time can be approximated by a second-degree polynomial. Given the factorial nature of the problem, with the number of possible solutions increasing in factorial scale in relation to the number of cities, this tendency of quadratic growth is good.

*6.2. Assessing the Quality of the Solutions.* For each of the 28 instances, the algorithm was run 30 times. The results obtained are presented in Table 4. For each instance, the result is presented with the following attributes: (1) the cost of the best solution found (Best); (2) the number of times the best-known solution (BKS) was found; (3) the average cost of the best solution found; (4) the percentage deviation of the best solution found for BKS (PDb); (5) the percentage deviation of the mean solution for BKS (PDav); (6) the average number of iterations for the convergence; (7) the average number of bees in the final swarm; and (8) the average time (Time), in seconds, of the algorithm execution.

Evaluating the best solution obtained for each of the instances in the 30 runs (Best and PDb in Table 4), TSPoptBees was able to find the best-known solution at least once for 19 of the 28 instances of TSP. Out of these 19 instances, the TSP was found more than once for 17 of them. Among the 9 instances for which the BKS was not found, the highest value for PDb was 0.44% for the pr439 instance. On average, the best solution found by TSPoptBees deviates by 0.08% from the BKS (average value for PDb).

Regarding the average solution obtained during the 30 runs for each instance (PDav and PDb in Table 4), TSPoptBees was not able to find the BKS in all runs for any instance.

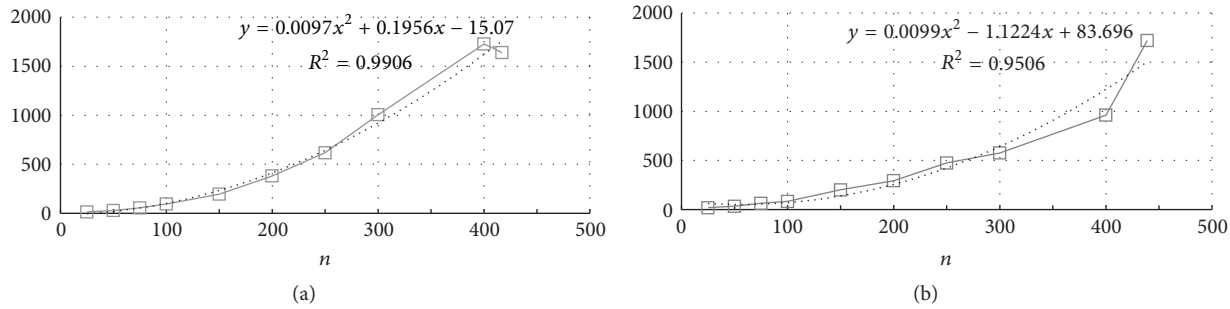



Figure 4: Running time of TSPoptBees for two TSP instances: (a) fl417 and (b) pr439.

Table 4: Computational results of TSPoptBees.

| Instance | BKS | Best | # BKS | Average | PDb | PDav | Iterations | # bees | Time |
|---|---|---|---|---|---|---|---|---|---|
| att48 | **10628** | 10628 | 15 | 10646.20 | 0.00 | 0.17 | 2159.50 | 180.40 | 49.45 |
| eil51 | **426** | 427 | 0 | 427.37 | 0.23 | 0.32 | 1954.63 | 137.87 | 40.13 |
| berlin52 | **7542** | 7542 | 29 | 7545.83 | 0.00 | 0.05 | 1486.20 | 161.33 | 33.56 |
| st70 | **675** | 675 | 8 | 679.73 | 0.00 | 0.70 | 2270.53 | 186.27 | 78.49 |
| eil76 | **538** | 538 | 12 | 541.20 | 0.00 | 0.59 | 3322.10 | 168.67 | 98.24 |
| pr76 | **108159** | 108159 | 23 | 108368.90 | 0.00 | 0.19 | 2426.93 | 192.13 | 81.91 |
| kroA100 | **21282** | 21282 | 16 | 21296.63 | 0.00 | 0.07 | 2061.30 | 219.27 | 86.89 |
| kroB100 | **22141** | 22141 | 17 | 22176.93 | 0.00 | 0.16 | 2575.17 | 225.13 | 123.98 |
| kroC100 | **20749** | 20749 | 16 | 20797.33 | 0.00 | 0.23 | 2706.03 | 223.67 | 132.27 |
| kroD100 | **21294** | 21294 | 3 | 21456.37 | 0.00 | 0.76 | 2992.10 | 238.33 | 139.68 |
| kroE100 | **22068** | 22106 | 0 | 22144.40 | 0.17 | 0.35 | 2627.53 | 226.60 | 122.91 |
| rd100 | **7910** | 7910 | 3 | 7997.07 | 0.00 | 1.10 | 2557.03 | 214.13 | 132.98 |
| eil101 | **629** | 629 | 8 | 630.93 | 0.00 | 0.31 | 2606.13 | 179.67 | 91.90 |
| lin105 | **14379** | 14379 | 29 | 14380.23 | 0.00 | 0.01 | 2352.03 | 250.80 | 129.81 |
| pr107 | **44303** | 44303 | 18 | 44345.60 | 0.00 | 0.10 | 1853.17 | 319.73 | 183.47 |
| pr124 | **59030** | 59030 | 20 | 59107.07 | 0.00 | 0.13 | 2296.20 | 309.47 | 181.00 |
| bier127 | **118282** | 118282 | 4 | 118650.20 | 0.00 | 0.31 | 2901.80 | 167.20 | 120.26 |
| pr136 | **96772** | 96785 | 0 | 97671.70 | 0.01 | 0.93 | 4030.93 | 225.13 | 275.90 |
| kroA150 | **26524** | 26583 | 0 | 26827.27 | 0.22 | 1.14 | 3586.33 | 243.47 | 291.64 |
| kroB150 | **26130** | 26130 | 3 | 26237.90 | 0.00 | 0.41 | 4075.33 | 242.00 | 371.65 |
| rat195 | **2323** | 2331 | 0 | 2352.53 | 0.34 | 1.27 | 3724.30 | 247.87 | 538.12 |
| kroA200 | **29368** | 29368 | 1 | 29527.93 | 0.00 | 0.54 | 4063.77 | 282.33 | 564.80 |
| kroB200 | **29437** | 29489 | 0 | 29771.77 | 0.18 | 1.14 | 4489.53 | 282.33 | 464.05 |
| tsp225 | **3916** | 3916 | 1 | 3984.37 | 0.00 | 1.75 | 4743.40 | 254.47 | 711.53 |
| a280 | **2579** | 2579 | 3 | 2613.50 | 0.00 | 1.34 | 4424.30 | 268.40 | 939.57 |
| lin318 | **42029** | 42196 | 0 | 42589.33 | 0.40 | 1.33 | 6341.37 | 267.67 | 1428.00 |
| fl417 | **11861** | 11902 | 0 | 12003.13 | 0.35 | 1.20 | 5907.13 | 492.80 | 3026.35 |
| pr439 | **107217** | 107685 | 0 | 110352.80 | 0.44 | 2.92 | 6524.4 | 299.20 | 2651.69 |

The largest deviation of the mean solution from the BKS was 2.92% for the pr439 instance. The average solution obtained by TSPoptBees for the set of instances used in this test had a deviation of 0.70% for the BKS (mean value for PDb).

Table 5 presents a comparison of the quality of the solutions obtained by TSPoptBees with those published in works that relate the TSP solution with algorithms inspired by bee behavior. The algorithms listed in this table are (1) BS + 2opt [82, 83], (2) BCO [36], (3) ABC-GSX [34], (4) ABC & PR [47], (5) CABC [45], (6) BCOPR [81], and (7) ABC + 2opt [60]. The best solutions in the comparison are shown in italic in the table and a dash (—) appears when the result is not published for that specific instance.

The HBMO-TSP uses in internal steps two techniques that already present, by themselves, competitive results for the TSP: (1) the heuristic called MPNS-GRASP (Marinakis et al., 2009) for the definition of the initial solutions and (2) the heuristic called ENS [97] for the local search stage. The BCO does not present any additional peculiarity to the structure of the algorithm itself that would favor the presented results.

Table 5: Comparison of the computational results of TSPoptBees with other bee-inspired algorithms from the literature.

| Instance | TSPoptBees | | BS + 2opt | | BCO | | ABC-GSX | | ABC & PR | | CABC | | BCOPR | | ABC + 2opt | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PDb | PDav | PDb | PDav | PDb | PDav | PDb | PDav | PDb | PDav | PDb | PDav | PDb | PDav | PDb | PDav |
| att48 | *0.00* | *0.17* | — | — | 0.31 | 0.83 | — | — | — | — | — | — | — | — | — | — |
| eil51 | 0.23 | 0.32 | 0.53 | 1.14 | 0.47 | 0.85 | *0.00* | 0.94 | — | *0.00* | — | — | *0.00* | 1.1 | *0.00* | 0.39 |
| berlin52 | *0.00* | 0.05 | *0.00* | 1.19 | — | — | *0.00* | 0.03 | — | — | — | — | — | — | *0.00* | *0.00* |
| st70 | *0.00* | 0.70 | 0.22 | 1.06 | — | — | *0.00* | 0.71 | — | *0.00* | — | — | 0.74 | 2.46 | *0.00* | 0.56 |
| eil76 | *0.00* | 0.59 | — | — | 0.19 | 2.01 | *0.00* | 2.41 | — | *0.00* | — | — | — | — | — | — |
| pr76 | *0.00* | *0.19* | 0.58 | 1.19 | — | — | *0.00* | 0.46 | — | — | — | — | — | — | *0.00* | 0.45 |
| kroA100 | *0.00* | *0.07* | 0.73 | 1.36 | 2.26 | 3.43 | *0.00* | 0.20 | — | — | — | — | — | — | *0.00* | 1.04 |
| kroB100 | *0.00* | *0.16* | — | — | 2.24 | 3.10 | — | — | — | — | — | — | — | — | — | — |
| kroC100 | *0.00* | *0.23* | — | — | 0.50 | 1.50 | — | — | — | — | — | — | — | — | — | — |
| kroD100 | *0.00* | *0.76* | — | — | 1.64 | 3.25 | — | — | — | — | — | — | — | — | — | — |
| kroE100 | *0.17* | *0.35* | — | — | 1.73 | 2.20 | — | — | — | — | — | — | — | — | — | — |
| rd100 | *0.00* | *1.10* | — | — | — | — | — | — | — | 1.12 | — | — | — | — | — | — |
| eil101 | *0.00* | *0.31* | 0.35 | 3.97 | 0.95 | 2.29 | — | — | — | 1.56 | — | — | 1.27 | 3.31 | 1.50 | 2.90 |
| lin105 | *0.00* | *0.01* | — | — | 0.32 | 1.24 | — | — | — | 1.24 | — | — | — | — | — | — |
| pr107 | *0.00* | *0.10* | — | — | — | — | — | — | — | 0.94 | — | — | — | — | — | — |
| pr124 | *0.00* | *0.13* | — | — | — | — | — | — | — | 0.36 | — | — | — | — | — | — |
| bier127 | *0.00* | *0.31* | — | — | — | — | — | — | — | 1.45 | — | — | — | — | — | — |
| pr136 | *0.01* | *0.93* | — | — | — | — | — | — | — | 1.56 | — | — | — | — | — | — |
| kroA150 | *0.22* | *1.14* | — | — | 5.03 | 6.39 | — | — | — | — | — | — | — | — | — | — |
| kroB150 | *0.00* | *0.41* | — | — | 1.55 | 3.68 | — | — | — | — | 0.45 | 0.98 | — | — | — | — |
| rat195 | *0.34* | *1.27* | — | — | — | — | — | — | — | 2.30 | — | — | — | — | — | — |
| kroA200 | *0.00* | *0.54* | — | — | 2.02 | 4.26 | — | — | — | — | 0.18 | 0.62 | — | — | — | — |
| kroB200 | *0.18* | *1.14* | — | — | 3.10 | 6.36 | — | — | — | — | — | — | — | — | — | — |
| tsp225 | *0.00* | *1.75* | 5.35 | 6.60 | — | — | — | — | — | — | — | — | — | — | 8.51 | 12.41 |
| a280 | *0.00* | *1.34* | 5.95 | 7.66 | — | — | — | — | — | — | — | — | 3.95 | 5.30 | 21.83 | 23.92 |
| lin318 | *0.40* | *1.33* | — | — | 6.32 | 7.55 | — | — | — | — | — | — | — | — | — | — |

## 7. Conclusions and Future Trends

Vehicle routing problems are of great importance in scientific research, since they present a wide set of practical applications and are difficult to solve in the optimality by exact methods, being one of the most studied classes of combinatorial optimization problems. Finding optimal solutions to these problems is unfeasible for large instances due to the factorial growth in the number of possible solutions. Thus, optimal solution methods are replaced by heuristics capable of finding good solutions in reasonable time. Among the many heuristics that have been applied to such problems, swarm-based algorithms, including bee-inspired methods, have received a great deal of attention over the past decade.

Due to the relevance of vehicle routing problems and swarm intelligence algorithms, the first contribution of this paper was to provide a thorough review of bee-inspired methods designed to solve this class of discrete optimization tasks. A taxonomy of methods was presented and the review followed in chronological order based on the taxonomy used and taking into account the problem solved and modifications introduced in the algorithm. It then followed with a presentation of the TSPoptBees algorithm for the solution of the most well-known vehicle routing problem: the travelling salesman problem. The proposed approach presented competitive results, obtaining, in average, solutions of better quality when compared to other algorithms inspired by the behavior of bees.

The survey presented showed that the Artificial Bee Colony (ABC) algorithm is by far the most explored proposal in the literature, but for obtaining good results in vehicle routing problems it invariably requires modifications, hybridizations, and improvements in its standard structure. The same holds true for the BCO, the Bee System (BS), and the MHBO algorithms. In the particular case of TSPoptBees, its former algorithm (optBees) was introduced to solve continuous optimization tasks and, thus, it had to be adapted to solve discrete optimization problems.

For virtually all bee-inspired algorithms, a number of issues open avenues for future research in the context of combinatorial optimization problems and others, such as methods to automatically define the input parameters, including self-adaptive approaches; the use of initialization, selection, and local search techniques specially designed for combinatorial optimization problems; the investigation and improvements of convergence times and properties; empirical studies about the dynamics of the swarm; the design of a general-purpose framework for bee-inspired

algorithms; the inclusion of diversity control mechanisms; a better understanding of the classes of problems these algorithms perform better and why; and the parallelization of the methods. Last but not least, bee-inspired approaches have a great potential to be adapted and applied to various other contexts, from pattern recognition to autonomous navigation.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

The authors would like to thank FAPESP, CNPq, Capes, and MackPesquisa for the financial support. As a Machine Learning Center of Excellence, special thanks also go to Intel for all its support.

## References

[1] S. Luke, *Essentials of Metaheuristics*, Lulu, 2nd edition, 2013.

[2] Z. Michalewicz and D. Fogel, *How to Solve It: Modern Heuristics*, Springer Science & Business Media, 2013.

[3] A. G. Cunha, R. Takahashi, and C. H. Antunes, *Evolutionary Computing and Metaheuristics Manual*, Imprensa da Universidade de Coimbra, Coimbra, Portugal, 2012.

[4] C. B. da Cunha, "Practical aspects of the application of vehicle routing models to real world problems," *Transportes*, vol. 8, no. 2, pp. 1–23, 2000.

[5] E. Lawler, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, Incorporated, 1985.

[6] 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.

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

[8] P. Toth and D. Vigo, *The Vehicle Routing Problem*, Society for Industrial and Applied Mathematics (SIAM), 2001.

[9] G. Gutin and A. Punnen, *The Traveling Salesman Problem And Its Variations*, Springer Science & Business Media, 2002.

[10] G. Laporte, "Fifty years of vehicle routing," *Transportation Science*, vol. 43, no. 4, pp. 408–416, 2009.

[11] G. Laporte, M. Gendreau, J. Potvin, and F. Semet, "Classical and modern heuristics for the vehicle routing problem," *International Transactions in Operational Research*, vol. 7, no. 4-5, pp. 285–300, 2000.

[12] P. Toth and D. Vigo, *Vehicle Routing*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[13] L. N. de Castro, *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications and Applications*, CRC Press, 2006.

[14] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2010.

[15] H. Zang, S. Zhang, and K. Hapeshi, "A review of nature-inspired algorithms," *Journal of Bionic Engineering*, vol. 7, pp. S232–S237, 2010.

[16] L. N. de Castro, "Fundamentals of natural computing: an overview," *Physics of Life Reviews*, vol. 4, no. 1, pp. 1–36, 2007.

[17] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems. [S.l.]*, Oxford University Press, 1999.

[18] C. Blum and D. Merkle, *Swarm Intelligence: Introduction and Applications, Natural Computing Series*, Springer, 2008.

[19] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[20] R. Eberhart and Y. Shi, "Particle Swarm Optimization: Developments, Applications and Resources," in *Proceedings of the Congress on Evolutionary Computation, IEEE 2001*, pp. 81–86, 2001.

[21] R. D. Maia, L. N. de Castro, and W. M. Caminhas, "Collective decision-making by bee colonies as model for optimization - the OptBees algorithm," *Applied Mathematical Sciences*, vol. 7, no. 85-88, pp. 4327–4351, 2013.

[22] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, vol. 31, no. 1–4, pp. 61–85, 2009.

[23] S. Bitam, M. Batouche, and E.-G. Talbi, "A survey on bee colony algorithms," in *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW '10)*, pp. 1–8, IEEE, New York, NY, USA, April 2010.

[24] J. A. Ruiz-Vanoye, O. Díaz-Parra, F. Cocón et al., "Meta-heuristics algorithms based on the grouping of animals by social behavior for the traveling salesman problem," *International Journal of Combinatorial Optimization Problems and Informatics*, vol. 3, no. 3, pp. 104–123, 2012.

[25] B. K. Verma and D. Kumar, "A review on artificial bee colony algorithm," *International Journal of Engineering & Technology*, vol. 2, no. 3, pp. 175–186, 2013.

[26] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, pp. 21–57, 2014.

[27] D. Agarwal, A. Gupta, and P. K. Singh, "A systematic review on artificial bee colony optimization technique," *International Journal of Control Theory and Applications*, vol. 9, no. 11, pp. 5487–5500, 2016.

[28] M. M. Millonas, "Swarms, Phase Transitions, and Collective Intelligence," in *LANGTON, C. G. Artificial Life III: Proceedings of the Workshop on Artificial Life*, pp. 417–445, LANGTON, C. G. Artificial Life III: Proceedings of the Workshop on Artificial Life, 1994.

[29] D. S. Johnson and A. McGeoch, "The traveling salesman problem: A case study in local optimization," *Local search in Combinatorial Optimization*, vol. 1, pp. 215–310, 1997.

[30] R. D. Angel, W. L. Caudle, R. Noonan, and A. Whinston, "Computer-assisted school bus scheduling," *Management Science*, vol. 18, no. 6, pp. 279–288, 1972.

[31] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[32] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. tr06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005, vol. 200.

[33] B. Basturk and D. Karaboga, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 687–697, 2008.

[34] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul, "ABC-GSX: A hybrid method for solving the traveling salesman problem," in *Proceedings of the 2010 2nd World Congress on Nature and Biologically Inspired Computing, NaBIC 2010*, pp. 7–12, jpn, December 2010.

[35] S. Nakrani and C. Tovey, "On honey bees and dynamic server allocation in internet hosting centers," *Adaptive Behavior*, vol. 12, no. 3-4, pp. 223–240, 2004.

[36] L. P. Wong, M. Y. H. Low, and C. S. Chong, "A bee colony optimization algorithm for traveling salesman problem," in *Proceedings of the Second Asia International Conference on Modelling & Simulation*, pp. 818–823, 2008.

[37] X. Lu and Y. Zhou, "A novel global convergence algorithm: bee collecting pollen algorithm," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence. ICIC 2008. Lecture Notes in Computer Science*, D. S. Huang, D. C. Wunsch, D. S. Levine, and Jo. K. H., Eds., vol. 5227, Springer, Berlin, Heidelberg, 2008.

[38] P. Lučić and D. Teodorović, "Bee System: Modeling Combinatorial Optimization Transportation Engineering Problems by Swarm Intelligence," in *Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis*, pp. 441–445, 2001.

[39] P. Lučić and D. Teodorović, "Transportation modeling: An artificial life approach," *Proceedings of the International Conference on Tools with Artificial Intelligence*, pp. 216–223, 2002.

[40] D. Teodorovic, P. Lucic, G. Markovic, and M. Dell'Orco, "Bee colony optimization: principles and applications," in *In Neural Network Applications in Electrical Engineering*, pp. 151–156, NEUREL, 2006.

[41] H. Abbass, "MHBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach," in *Proceedings of the, 2001 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 207–214, 2001.

[42] C. Yang, J. Chen, and X. Tu, "Algorithm of marriage in honey bees optimization based on the nelder-mead method," in *Proceedings of the International Conference on Intelligent Systems and Knowledge Engineering (ISKE 2007). Advances in Intelligent Systems Research*, 2007.

[43] C. Yang, X. Tu, and J. Chen, "Algorithm of marriage in honey bees optimization based on the wolf pack search," in *Proceedings of the 2007 International Conference on Intelligent Pervasive Computing, IPC 2007*, pp. 462–467, kor, October 2007.

[44] Y. Marinakis, M. Marinaki, and G. Dounias, "Honey bees mating optimization algorithm for the vehicle routing problem," *Studies in Computational Intelligence*, vol. 129, pp. 139–148, 2008.

[45] 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.

[46] V. Singh, R. Tiwari, D. Singh, and A. Shukla, "RGBCA-genetic bee colony algorithm for travelling salesman problem," in *Proceedings of the 2011 World Congress on Information and Communication Technologies, WICT 2011*, pp. 1002–1008, ind, December 2011.

[47] X. Zhang, Q. Bai, and X. Yun, "A new hybrid artificial bee colony algorithm for the traveling salesman problem," in *Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks, ICCSN 2011*, pp. 155–159, chn, May 2011.

[48] W. Y. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 215, no. 1, pp. 126–135, 2011.

[49] E. Özceylan, M. S. Kiran, and Y. Atasagun, "A new hybrid heuristic approach for solving green traveling salesman problem," in *Proceedings of the 41st International Conference on Computers and Industrial Engineering*, pp. 23–26, 2011.

[50] P. Ji and Y. Wu, "An improved artificial bee colony algorithm for the capacitated vehicle routing problem with time-dependent travel times," in *Proceedings of the Tenth International Symposium On Operations Research And Its Applications (ISORA 2011*, pp. 75–82, 2011.

[51] W. Li, W. Li, Y. Yang, H. Liao, J. Li, and X. Zheng, "Artificial bee colony algorithm for traveling salesman problem," *Advanced Materials Research*, vol. 314-316, pp. 2191–2196, 2011.

[52] I. Brajevic, "Artificial bee colony algorithm for the capacitated vehicle routing problem," in *Proceedings of the European Computing Conference, ECC '11*, pp. 239–244, 2011.

[53] Y.-J. Shi, F.-W. Meng, and G.-J. Shen, "A modified artificial bee colony algorithm for vehicle routing problems with time windows," *Information Technology Journal*, vol. 11, no. 10, pp. 1490–1495, 2012.

[54] S. Iqbal and M. S. Rahman, "Vehicle routing problems with soft time windows," in *Proceedings of the 2012 7th International Conference on Electrical and Computer Engineering, ICECE 2012*, pp. 634–638, 2012.

[55] K. Karabulut and M. F. Tasgetiren, "A discrete artificial bee colony algorithm for the traveling salesman problem with time windows," in *Proceedings of the 2012 IEEE Congress on Evolutionary Computation, CEC 2012*, aus, June 2012.

[56] A. S. Bhagade and P. V. Puranik, "Artificial Bee Colony (ABC) Algorithm For Vehicle Routing Optimization Problem," *International Journal of Soft Computing and Engineering*, vol. 2, pp. 329–333, 2012.

[57] N. Pathak and S. P. Tiwari, "Traveling Salesman Problem Using Bee Colony with SPV," *International Journal of Soft Computing and Engineering*, vol. 2, no. 3, pp. 410–414, 2012.

[58] L. Li, Y. Cheng, L. Tan, and B. Niu, "A discrete artificial bee colony algorithm for TSP problem," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6840, pp. 566–573, 2012.

[59] W. Bin, C. Hong, and Z.-Y. Cui, "Artificial bee colony algorithm for two-dimensional loading capacitated vehicle routing problem," in *Proceedings of the 2013 20th International Conference on Management Science and Engineering, ICMSE 2013*, pp. 406–412, chn, July 2013.

[60] M. S. Kiran, H. Işcan, and M. Gündüz, "The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem," *Neural Computing and Applications*, vol. 23, no. 1, pp. 9–21, 2013.

[61] S. Sabet, F. Farokhi, and M. Shokouhifar, "A hybrid mutation-based artificial bee colony for traveling salesman problem," *Lecture Notes on Information Theory*, vol. 1, no. 3, pp. 99–103, 2013.

[62] B. Yao, P. Hu, M. Zhang, and S. Wang, "Artificial bee colony algorithm with scanning strategy for the periodic vehicle routing problem," *Simulation*, vol. 89, no. 6, pp. 762–770, 2013.

[63] C.-Y. Liu, "An improved adaptive genetic algorithm for the multi-depot vehicle routing problem with time window," *Journal of Networks*, vol. 8, no. 5, pp. 1035–1042, 2013.

[64] W. Yang and Z. Pei, "Hybrid ABC/PSO to solve travelling salesman problem," *International Journal of Computing Science and Mathematics*, vol. 4, no. 3, pp. 214–221, 2013.

[65] S. Pandey and S. Kumar, "Enhanced artificial bee colony algorithm and its application to travelling salesman problem," *HCTL Open International Journal of Technology Innovations and Research*, vol. 2, pp. 137–146, 2013.

[66] A. Rekaby, A. A. Youssif, and A. Sharaf Eldin, "Introducing adaptive artificial bee colony algorithm and using it in solving traveling salesman problem," in *Proceedings of the Science and Information Conference (SAI '13)*, pp. 502–506, 2013.

[67] Y. Yuan and Y. Zhu, "A hybrid artificial bee colony optimization algorithm," in *Proceedings of the 2014 10th International Conference on Natural Computation, ICNC 2014*, pp. 492–496, chn, August 2014.

[68] S. Zhang, C. K. M. Lee, K. L. Choy, W. Ho, and W. H. Ip, "Design and development of a hybrid artificial bee colony algorithm for the environmental vehicle routing problem," *Transportation Research Part D: Transport and Environment*, vol. 31, pp. 85–99, 2014.

[69] O. E. Nahum, Y. Hadas, and U. Spiegel, "Multi-objective vehicle routing problems with time windows: A vector evaluated artificial bee colony approach," *International Journal of Computer and Information Technology*, vol. 3, pp. 41–47, 2014.

[70] H. E. Kocer and M. R. Akca, "An improved artificial bee colony algorithm with local search for traveling salesman problem," *Cybernetics and Systems*, vol. 45, no. 8, pp. 635–649, 2014.

[71] Y.-T. Chung, "A Hybrid Artificial Bee Colony Algorithm to Solve the Capacitated Vehicle Routing Problem in Logistics Management, Master's Dissertation," in *Department of Industrial Management*, p. 81, National Taiwan University of Science and Technology, 2014.

[72] R. Nagaya and A. Inoie, "Hybrid ABC algorithm for the capacitated vehicle routing problem," in *Proceedings of the 8th International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 383, 382 pages, 2014.

[73] S. Z. Zhang and C. K. M. Lee, "An Improved Artificial Bee Colony Algorithm for the Capacitated Vehicle Routing Problem," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, SMC 2015*, pp. 2124–2128, hkg, October 2015.

[74] M. Gündüz, M. S. Kiran, and E. Özceylan, "A hierarchic approach based on swarm intelligence to solve the traveling salesman problem," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 23, no. 1, pp. 103–117, 2015.

[75] M. Alzaqebah, S. Abdullah, and S. Jawarneh, "Modified artificial bee colony for the vehicle routing problems with time windows," *SpringerPlus*, vol. 5, no. 1, article no. 1298, 2016.

[76] L. P. Wong, M. Y. Hean Low, and C. S. Chong, "An efficient bee colony optimization algorithm for traveling salesman problem using frequency-based pruning," in *Proceedings of the 7th IEEE International Conference on Industrial Informatics (INDIN '09)*, pp. 775–782, IEEE, Wales, UK, June 2009.

[77] L. P. Wong, M. Y. H. Low, and C. S. Chong, "A bee colony optimization algorithm with the fragmentation state transition rule for traveling salesman problem," in *Proceedings of the 2009 Conference on Innovative Production Machines and Systems (IPROMS 2009)*, pp. 399–404, 2009.

[78] L.-P. Wong, M. Y. Hean Low, and C. S. Chong, "A generic bee colony optimization framework for combinatorial optimization problems," in *Proceedings of the Asia Modelling Symposium 2010: 4th International Conference on Mathematical Modelling and Computer Simulation, AMS2010*, pp. 144–151, 2010.

[79] L.-P. Wong, M. Y. H. Low, and C. S. Chong, "Bee colony optimization with local search for traveling salesman problem," *International Journal on Artificial Intelligence Tools*, vol. 19, no. 3, pp. 305–334, 2010.

[80] S. Häckel and P. Dippold, "The Bee Colony-inspired Algorithm (BCiA): A two-stage approach for solving the vehicle routing problem with time windows," in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference, GECCO-2009*, pp. 25–32, can, July 2009.

[81] A. S. Girsang, C.-W. Tsai, and C.-S. Yang, "A fast bee colony optimization for traveling salesman problem," in *Proceedings of the 3rd International Conference on Innovations in Bio-Inspired Computing and Applications, IBICA 2012*, pp. 7–12, 2012.

[82] P. Lučić and D. Teodorović, "Computing with bees: Attacking complex transportation engineering problems," *International Journal on Artificial Intelligence Tools*, vol. 12, no. 03, pp. 375–394, 2003.

[83] P. Lučić and D. Teodorović, "Vehicle routing problem with uncertain demand at nodes: the bee system and fuzzy logic approach," in *Fuzzy Sets Based Heuristics for Optimization*, vol. 126 of *Studies in Fuzziness and Soft Computing*, pp. 67–82, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[84] M. Nikolić, D. Teodorović, and M. Šelmić, "Solving the vehicle routing problem with time windows by bee colony optimization metaheuristic," in *In Proceedings of the 1st Logistics International Conference*, pp. 44–48, 2013.

[85] S. Jawarneh and S. Abdullah, "Sequential insertion heuristic with adaptive bee colony optimisation algorithm for vehicle routing problem with time windows," *PLoS ONE*, vol. 10, no. 7, 2015.

[86] 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.

[87] Y. Celik and E. Ulker, "A marriage in honey bee optimisation approach to the asymmetric travelling salesman problem," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 6, pp. 4123–4132, 2012.

[88] Q. Ruan, Z. Zhang, L. Miao, and H. Shen, "A hybrid approach for the vehicle routing problem with three-dimensional loading constraints," *Computers & Operations Research*, vol. 40, no. 6, pp. 1579–1589, 2013.

[89] A. Singh and D. Narayan, "Augmentation of travelling salesman problem using bee colony optimization," *International Journal of Innovative Technology and Exploring Engineering*, 2012.

[90] R. D. Maia, L. N. De Castro, and W. M. Caminhas, "Bee colonies as model for multimodal continuous optimization: The OptBees algorithm," in *Proceedings of the 2012 IEEE Congress on Evolutionary Computation, CEC 2012*, aus, June 2012.

[91] D. P. F. Cruz, R. D. Maia, A. Szabo, and L. N. de Castro, "A bee-inspired algorithm for optimal data clustering," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation, CEC 2013*, pp. 3140–3147, mex, June 2013.

[92] D. P. F. Cruz, R. D. Maia, L. A. da Silva, and L. N. de Castro, "BeeRBF: A bee-inspired data clustering approach to design RBF neural network classifiers," *Neurocomputing*, vol. 172, Article ID 15835, pp. 427–437, 2016.

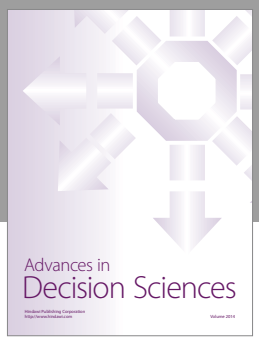
[93] T. A. S. Masutti and L. N. De Castro, "TSPoptBees: A bee-inspired algorithm to solve the traveling salesman problem," in *Proceedings of the 5th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2016*, pp. 593–598, jpn, July 2016.

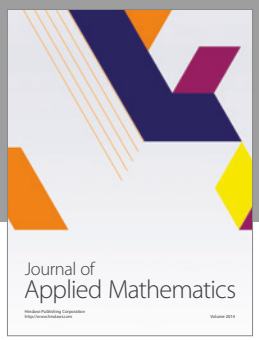
[94] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

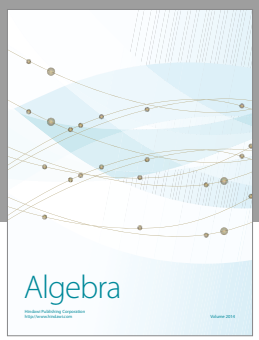
[95] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

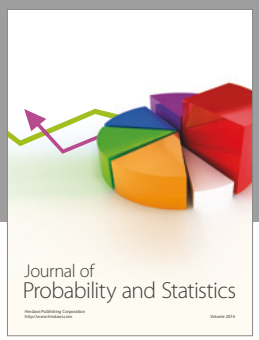
[96] T. A. S. Masutti and L. N. de Castro, "Parameter analysis of a bee-inspired algorithm to solve the traveling salesman problem," in *Proceedings of the 15th IASTED International Conference on Intelligent Systems and Control, ISC 2016*, pp. 245–252, 2016.

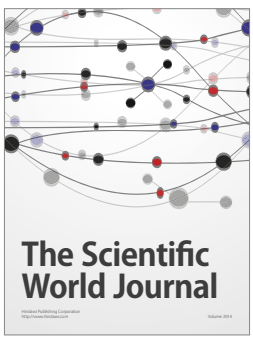
[97] Y. Marinakis, A. Migdalas, and P. M. Pardalos, "Expanding neighborhood GRASP for the traveling salesman problem," *Computational Optimization and Applications*, vol. 32, no. 3, pp. 231–257, 2005.

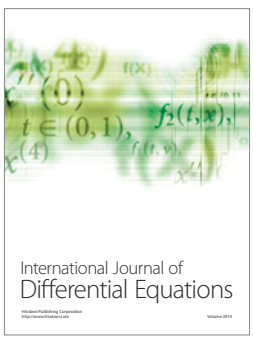


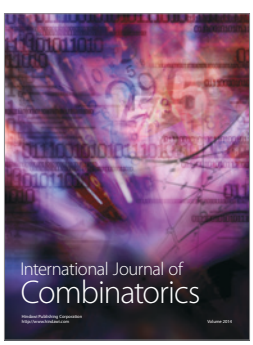


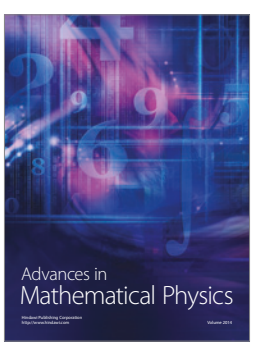


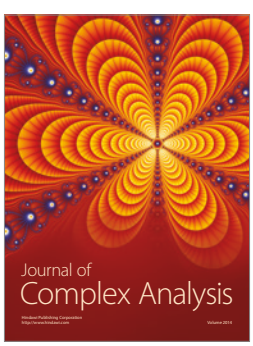


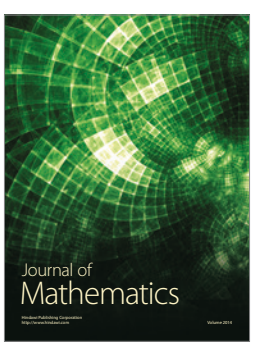


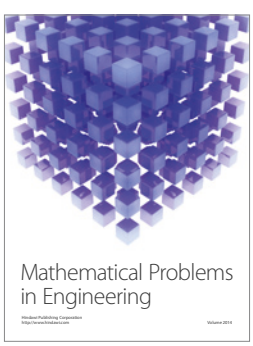


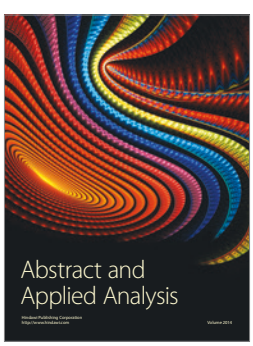


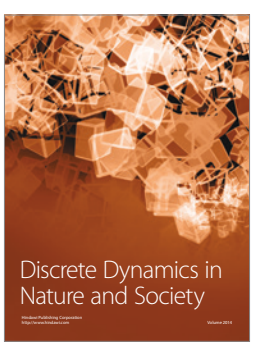


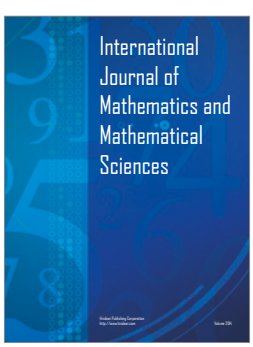


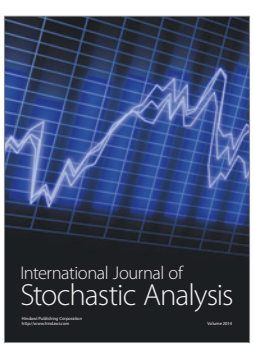












Hindawi

Submit your manuscripts at
https://www.hindawi.com