The Bee Colony-inspired Algorithm (BCiA) – A Two-Stage Approach for Solving the Vehicle Routing Problem With Time Windows

Sascha Häckel
Faculty of Economics and Business Administration
Chemnitz University of Technology
Chemnitz, Germany
shae@hrz.tu-chemnitz.de

Patrick Dippold
Faculty of Economics and Business Administration
Chemnitz University of Technology
Chemnitz, Germany
padi@hrz.tu-chemnitz.de

ABSTRACT

This article presents a new optimization algorithm, which adapts the behavior of honey bees during their search for nectar. In addition to the established ant algorithms, bee-inspired algorithms represent a relatively young form of solution procedures, whose applicability to the solution of complex optimization problems has already been shown.

The herein presented two-stage approach belongs to the class of metaheuristics to control a construction heuristic and has been applied successfully to the NP-hard Vehicle Routing Problem with Time Windows (VRPTW).

Within this article, evaluation results are presented, which compare the developed algorithm to some of the most successful procedures for the solution of benchmark problems. The pursued approach gives the best results so far for a metaheuristic to control a construction heuristic.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms, Design, Measurement, Verification

Keywords

Swarm intelligence, Multi-objective optimization, Vehicle Routing Problem, Combinatorial Optimization

1. INTRODUCTION

Vehicle Routing Problems represent a significant issue in Operations Research. This is induced not least by problems with an economic background as well as by the huge cost-saving potential involved. A typical and often discussed problem of this class is the standard Vehicle Routing Problem, expanded by time windows (VRPTW). Due to the enormous complexity to solve this NP-hard problem, it will normally have to be resorted to heuristic methods for an approximate solution. Especially nature-inspired metaheuristics like Ant Colony Optimization (ACO) or Evolutionary Algorithms (EA) often turn out to be extremely successful within literature. This article presents a new optimization algorithm for an application to the VRPTW, which is adapting the behavior of bees during their search for nectar.

In the first part of this paper, the discussed problem is presented and formalized. Furthermore, an overview is given on different successful algorithms for the VRPTW, which are presented in the literature. In the second part of the paper, different published algorithms that adapt the behavior of honey bees during their forage are discussed. Afterwards a new bee-inspired algorithm is presented, which has been applied successfully to the VRPTW. Finally, this will be shown on evaluation results compared to other methods.

GECCO'09, July 8–12, 2009, Montréal, Québec, Canada.

2. VRPTW: PROBLEM DESCRIPTION

In the following, the problem of the standard Vehicle Routing Problem as well as the problem with time windows is explained in general and formalized.

2.1 General Description

The objective of Vehicle Routing Problems is to efficiently group delivery or pick-up orders from geographically distributed customers to tours. The tours will be allocated to some means of transport (vehicle), each with a predefined maximum capacity. It is assumed that customer orders are consist of homogeneous goods that can be interpreted in capacity units.

In case of a Vehicle Routing Problem with Time Windows, additional static time intervals will be assigned to customers, within which the delivery by the vehicles has to take place. In a feasible solution of a VRPTW, all customer orders have to be achieved within the given time frames, without exceeding the maximum capacity.

In general, the objective of the standard Vehicle Routing Problem is to minimize the total tour length with a given (maximum) number of vehicles. The objective of the problem with time windows is, in contrast to the standard problem, almost exclusively described as a lexicographical target function within literature. The primary target of the objective function is the minimization of the total number of

vehicles needed, whereas the secondary target is to minimize the required total tour length.

However, these two targets are conflicting. The reason for this conflict is that, for the optimal solution with a minimal total number of vehicles, considerably more detours have to be made in contrast to the optimal solution with the single objective of a minimal total tour length and an inferior number of vehicles.

The Vehicle Routing Problem without time windows is already an NP-hard problem. This has been shown by a polynomial reduction of the TSP to the VRP by Lenstra and Rinnooy Kan in 1981 [18]. In 1985, Savelsbergh gave proof that time windows fundamentally complicate the problem solving [24]. Within the TSPTW, the search for an approved solution already represents an NP-hard problem, if the number of tours is smaller than the number of cities. This characteristic obviously does also exist concerning the VRPTW, if the number of customers is higher than the number of vehicles.

2.2 Formal Definition

The problem definition of the VRPTW is based on a directed graph $D = (V, E)$. The set of nodes V consists of set of customers $C \subset V$, with $i \in C$ and $i = 1, .., n$, and the two auxiliary nodes 0 and $n+1$, with $0, n+1 \notin C$, representing the depot. The set of directed edges E consists of all sorted pairs of customers $(i, j) \in C \times C$ with $i \neq j$ as well as the auxiliary edges $(0, i)$ and $(i, n+1)$. All edges are weighted with a spacious distance $d_{i,j} \geq 0$ and a temporal distance $t_{i,j} \geq 0$. Furthermore, there is a service time s_i as well as an interval $[a_i, b_i]$ assigned to each node i. The interval describes the earliest and the latest possible start of service at customer i.

Each tour of a vehicle k, with $k \in K$ out of the set of vehicles K, starts in node 0 and ends in node $n+1$. The assignment of customers to a tour occurs with decision variable $x^k_{i,j} \in \{0, 1\}$. $x^k_{i,j}$ takes value 1 if vehicle k serves customer j directly after customer i. The same applies for depot nodes. The service at customer j starts at time S^k_j. The arrival time $I^k_{i,j}$ at customer j directly following customer i within the tour results in $I^k_{i,j} = S^k_i + s_i + t_{i,j}$, whereas $S^k_0 = a_0$ is set for the depot. The start of service results in $S^k_j = max\{a_j, I^k_{i,j}\}$ with $i : \ x^k_{i,j} = 1$ accordingly.

There is a demand $q_i \geq 0$ assigned to each customer, while the demands of the depot nodes 0 and $n+1$ are set to 0. The vehicles are equipped with a maximum capacity Q.

The tours of all vehicles combined result in a route schedule φ. The problem can be described with equations (1)-(3) as follows. The lexicographic objective function $z(\varphi)$ of equation (1) is comprised of the objective function of the primary target to minimize the total number of vehicles in equation (2) and the objective function of the secondary target to minimize the overall tour length in equation (3).

$$lex \ \min \ z(\varphi) = \begin{pmatrix} f(\varphi) \\ g(\varphi) \end{pmatrix} \quad \text{with} \tag{1}$$

$$f(\varphi) = \sum_{k \in K} \sum_{j \in C} x^k_{0,j} \tag{2}$$

$$g(\varphi) = \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} d_{i,j} \cdot x^k_{i,j} \tag{3}$$

subject to

$$\sum_{k \in K} \sum_{j \in V} x^k_{i,j} = 1 \qquad i \in C \tag{4}$$

$$\sum_{i \in V} x^k_{i,c} - \sum_{j \in V} x^k_{c,j} = 0 \qquad k \in K, \ c \in C \tag{5}$$

$$\sum_{j \in V} x^k_{0,j} = \sum_{i \in V} x^k_{i,n+1} = 1 \qquad k \in K \tag{6}$$

$$\sum_{i \in C} q_i \cdot \sum_{j \in V} x^k_{i,j} \leq Q \qquad k \in K \tag{7}$$

$$x^k_{i,j} \cdot \left(\max \left(I^k_{i,j}, a_j \right) - S^k_j \right) = 0 \qquad \begin{matrix} (i,j) \in E, \\ \forall k \in K \end{matrix} \tag{8}$$

$$a_i \cdot \sum_{j \in V} x^k_{i,j} \leq S^k_i \qquad k \in K, \ i \in C \tag{9}$$

$$b_i \cdot \sum_{j \in V} x^k_{i,j} \geq S^k_i \qquad k \in K, \ i \in C \tag{10}$$

$$a_i \leq S^k_i \leq b_i \qquad \begin{matrix} \forall k \in K, \\ i \in \{0, n+1\} \end{matrix} \tag{11}$$

$$x^k_{i,j} \in \{0, 1\} \qquad \begin{matrix} \forall k \in K, \\ (i,j) \in E \end{matrix} \tag{12}$$

Equation (4) assures that each customer is served exactly once and by one vehicle only. Equations (5) and (6) control the construction of feasible tours, as each customer who has been visited by the relevant vehicle has to be left again and, furthermore, that each tour has to start in node 0 and to end in node $n+1$. Equation (8) describes the valid sequence of customers within the tours. Equation (7) ensures the compliance with the capacity limits of the vehicles. The constraints (9)-(11) enforce that the beginning of service at each visited customer or depot lies within the given time interval. Equation (12) describes the decision variable x.

3. AN OVERVIEW ON METAHEURISTICS FOR THE VRPTW

This section offers an overview on metaheuristic approaches for the VRPTW. As a large number of different approaches has been published within literature, first of all a classification of the different strategies is outlined, before some selected algorithms, which are known to be very successful, will be discussed briefly.

3.1 Classification

Homberger [14] classifies metaheuristics in type I and II. Type I represents metaheuristics to control a construction heuristic, whereas type II represents those metaheuristics to control a neighborhood search. Within literature, the majority of approaches are of type II, which also show better results than methods of type I. Metaheuristics can furthermore be differentiated into hybrid and non-hybrid methods, whereas hybrid methods use more than one metaheuristic steering principle. Successful methods of both classes can be found in the literature.

A further important classification criterion is given by the distinction between one-stage and two-stage strategies. Two-stage approaches consist of temporally successive phases. The first stage represents an independent metaheuristic approach for the optimization of the primary objective, the minimization of the number of vehicles used. Within the

following second stage, the total tour length is optimized as the secondary objective based on the results of stage I. Within both stages, there are mostly different target and fitness functions applied as well as different metaheuristic steering principles (hybrid). In case of metaheuristics of type II different problem-specific heuristics are applied very often. However, one-stage approaches pass on this bisection of the solution strategy but have not turned out to be that successful in the literature. Nearly all new methods are based on two-stage strategies (see 3.2).

3.2 Successful Algorithms

In the following, some selected and very successful approaches for the VRPTW will be presented. Overviews on further methods can be found in Bräysy et al. [5], [6]. All approaches presented and discussed herein are two-stage strategies and all methods apply different fitness functions, which are specifically designed either for the reduction of the number of vehicles or for that of the total tour length. Except for the two ACO-approaches in [13] and [12], all methods can be considered a metaheuristic of type II.

Homberger developed several successful hybrid methods using Tabu Search, Evaluation Strategies and Simulated Annealing for the control of neighborhood search methods [14]. Within each stage, he uses different metaheuristics. This is done by trying to insert the customers of the smallest tour into other tours an so reducing the total number of tours needed.

The method by Bent et al. uses Simulated Annealing in the first stage to control a neighborhood search [3]. Different operators are applied for the solution variation, which change customers or customer chains within or between the tours. In stage two, a *Large Neighborhood Search (LNS)* is applied. LNS belongs to the class of *partially-destructive/constructive*-methods, with which route schedules can first be destroyed partially by removing a certain number of customers on basis of a heuristics value and subsequently by joining them again by an insertion heuristic. This method realizes LNS as a local search.

Both methods by Pisinger et al. [22] and Prescott-Gagnon et al. [23] use different metaheuristics to control an *Adaptive Large Neighborhood Search (ALNS)*. Within the adaptive variant of LNS, several heuristics for a modification of a solution are applied. Out of these one method is chosen probabilistically according to related selection probabilities. After varying the solution, a fitness correlated adaption of the selection probabilities takes place.

Pisinger et al. use six different methods to remove customers from the tours and two methods for inserting them again. One method each is chosen probabilistically. By using a Simulated Annealing method, it is afterwards decided on the acceptance of the modified solution [22]. On the contrary, Prescott-Gagnon et al. use four different methods for removing customers, whereas the reinsertion is done with a heuristic Branch-and-Price method with an embedded Tabu Search [23].

Gambardella et al. present a non-hybrid metaheuristic of type I based on Ant Colony Optimization [13]. The so called *Multiple Ant Colony System (MACS-VRPTW)* comprises two stages, which are represented by separate but back-coupled ant colonies. The first colony *ACS-VEI* optimizes the target of reducing the number of vehicles required by searching for a solution with one vehicle less than the currently best known feasible solution. The second colony *ACS-TIME* optimizes the total tour length given a fixed number of vehicles, which is determined by the best feasible solution of *ACS-VEI* so far. There are several extensions available for this algorithm.

Ellabib et al. developed a parallel ant algorithm, also known as *Multiple Ant Colony System (M-ACS)* [12]. Using this method, several similar ant colonies are applied in parallel, each searching for a solution to the problem. Thereby, the best solutions of the colonies are exchanged between them and used for a pheromone update to integrate them into the optimization process of all colonies. Furthermore the method of Ellabib et al. uses a subsequent *2-Opt**-heuristic as a local search.

4. ADAPTING BEES BEHAVIOR

Within the following section, a new algorithm for the solution of the VRPTW will be presented. This algorithm is inspired by the behavior of honey bees during their forage. Therefore, it will first be dealt with this behavior, until afterwards the algorithm will be discussed in detail.

4.1 From Real To Artificial Bees

Although the majority of bees lives solitary, social bees like the honey bee shall be considered at this point. Those bees show all the characteristics of *eusociality*, which means „really social“:

- Overlapping of generations
- Separation into fertile and infertile caste
- Division of labor
- Brood care

In the following, the focus will be on the behavior of the bees during their *forage* and therewith the last phase in a bee's life. After the eclosion, the young bee primarily centers on tasks within the beehive as for example the brood care or the transport of nectar. Starting at about the 20th day of a bee's life until its death, the bee undertakes flights outside the hive.

Although the bee colony has a queen, no hierarchical but a decentral control exists. The beehive can instead be understood as a *self-organizing* system with a multiplicity of agents [17]. According to [4], a self-organizing system is based on characteristics of a positive and negative feedback, a random fluctuation as well as the interaction of the system's individuals. The use of preferably good food sources is an *emergent* property of the beehive.

Already Aristotle [1] observed the communication behavior of the bees but was unable to define it. An first explanation was given by the biologist Karl von Frisch [28] in the middle of the 20th century. The bees use a *dance language* inside the hive to communicate the location of the food sources. Sources which are located closer to the hive are solicited by a *round dance*. With this dance, the bees prescribe several circles with a changing orientation. Food sources with a further distance to the hive are communicated through a *waggle dance*. For this dance, the dancing bee starts to bounce with its abdomen. While doing this, the bee moves forward and finally returns in an arc of a circle to her starting point. This behavior is repeated several times. The duration and intensity of the waggle dance indicates the distance of the food source to the nest, while the orientation of the bee during the waggle phase shows the direction. In

contrast to the ants, bees use a direct form of communication.

After unloading the collected food a foraging bee returning to the beehive from a patch of flowers (*employed bee*) decides whether to abandon the food source or not. If the food source is abandoned, the bee observes the dances of other employed bees and follows one of the promoted ways as a *follower bee* or starts to search for an entirely new source as a *scout bee*. However, if the food source is not abandoned, the employed bee decides whether to dance for the source to recruit other bees with that or not. Camazine and Sneyd [7] developed the decision model presented in figure 1, with which the behavior of bees can be explained.

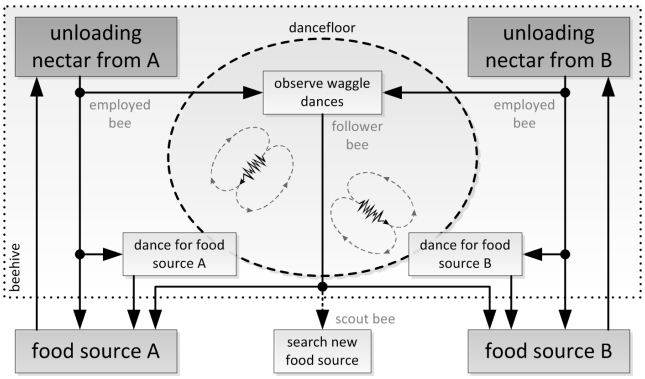

Figure 1: Decision model of the bees after [7]

The authors assume that solely the quality of the food sources is taken into consideration as a decision criterion. Furthermore, they assume that during the selection of the observed dance the follower bees do not compare the dances but follow the first randomly noticed dancing bee. Therewith, the selection rule of equation (13) is quite similar to the *Roulette Wheel Decision* used in most ant algorithms.

$$p_a = \frac{m_a \cdot td_a}{m_a \cdot td_a + m_b \cdot td_b} = 1 - p_b \qquad (13)$$

with:

$p_{a/b}$	selection probability for nectar source A/B
$m_{a/b}$	number of dancers for nectar source A/B
$td_{a/b}$	duration of the dance for nectar source A/B

4.2 Methods Inspired By Bees

Numerous algorithms adapting the behavior of bees have already been suggested in the literature. However, there has not been any comprehensive definition of a metaheuristic comparable to the ACO-metaheuristic by Dorigo/Stützle [11] in ant algorithms. Baykasoğlu et al. [2] distinguish the proposed algorithms by the adapted behavior of the bees. Besides those algorithms inspired by the forage of the bees, there are other algorithms based, for example, on the mating flight.

The two most inspiring bee algorithms for the development of the new method to solve the VRPTW were proposed by Karaboga [15], [16] and Pham et al. [21], respectively.

The *Artificial Bee Colony (ABC)* approach by Karaboga is based on a population of bees, of which one half are employed bees and the other half are of follower bees. Besides this, an employed bee can become a scout bee from time to time. There is only one employed bee assigned to each food source (solution to the considered problem). The algorithm starts with an initial population P of employed bees. Based on their solution quality, the probabilistic selection of the dances by the follower bees takes place according to equation (13). After that, a follower bee constructs its solution under consideration of the *preferred path*, which has been communicated by following the dance. Only the best solution of each food source (either the one of the employed bee or one of the follower bees) is taken over to the new population P′. An age-based strategy ensures that a solution, which cannot be improved for a certain number of iterations, is then removed from the population. It is replaced by the solution of a scout bee.

In the *Bees Algorithm (BA)* of Pham et al. also only one employed bee per solution is allowed. Based on a number of randomly created solutions, the z best ones are taken to form the initial population P_z of employed bees. The selection of dances by the follower bees is done deterministically. The e best employed bees of P_z each recruit a fixed number of follower bees. The other $(z-e)$ employed bees also recruit a fixed but smaller number of bees per iteration. After the solution construction of the follower bees, the best solution per food source is selected. Finally, the remaining bees randomly create additional solutions as scout bees. From this set of existing solutions, again the z best ones are selected and taken over into the new population P'_z.

With the *Honey Bee Colony Algorithm (HBCA)*, Chong et al. [8] developed an approach that is strongly influenced by ant algorithms. The follower bees decide for a preferred path randomly, which they then follow with a certain probability. By considering a heuristics value, the divergence does not take place uniformly distributed but in particular when the preferred edge has a low heuristics value. Not scout bees but an age-based replacement strategy is used to diversify the search. In [9], the bees are used to control a neighborhood search instead.

Lučić uses a completely different approach for the *Bee System (BS)* [19]. Here, the bees construct their solutions stepwise and also communicate about the partial solutions. Therefore an employed bee recruits follower bees for its respective partial solution. A recruited follower bee follows the preferred path exactly and adds additional solution components to the partial solution aftwerwards. This approach was generalized by Teodorović and Dell'Orco in [27] resulting in the *Bee Colony Optimization (BCO)* metaheuristic, which, however, cannot be put on the same level as the ACO-metaheuristic.

Other bee algorithms derive from Nakrani und Tovey [20] and Wedde et al. [29].

5. THE BEE COLONY-INSPIRED ALGORITHM

The following section will deal with the proposed bee-inspired algorithm for solving the VRPTW. It will be discussed in three consecutive steps: first, the overall principle of BCiA is presented. Afterwards, the process of solution construction by the bees in general and specifically applied to the VRPTW is described in more detail.

5.1 General Concept

With the *Bee Colony-inspired Algorithm (BCiA)*, a new method for the solution of the VRPTW has been developed,

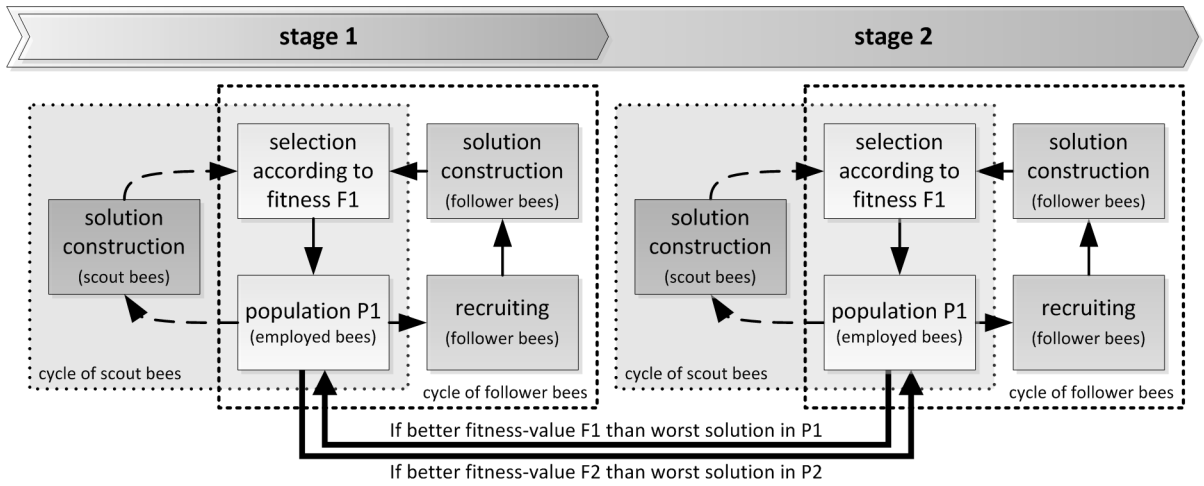

Figure 2: General Control Principle of the BCiA

which is based on the findings regarding formerly proposed successful solution procedures and follows the principles of bee algorithms. At present, this is the only application of a bee-inspired algorithm to the VRPTW.

The BCiA is based on a two-stage approach to avoid the existing conflict between the two objectives of the VRPTW. In the first stage the algorithm tries to continuously reduce the number of vehicles needed for the construction of feasible solutions. If this is not possible any longer, the algorithm proceeds to the second stage (minimizing total tour length). Furthermore, the BCiA is based on two populations P1 and P2, of which P1 is already active in the first stage and P2 engages in the second. Population P1 gets a fitness function F1 deviating from the target function of VRPTW, which is supposed to support the objective of reducing the number of tours. Subordinate to the number vehicles, the number of customers in the smallest tour ι of the route schedule φ is considered. If this number is equal for two comparable route schedules, with the „required total delay“, a value suggested by Homberger [14], is regarded in the third place. The total tour is consulted as a criterion for comparison only on the fourth level of F1. The fitness function F2 of population P2 corresponds to the objective function of the VRPTW. F1 as well as F2 are functions that have to be minimized.

The general steering principle of BCiA is illustrated in figure 2 and can also be seen in algorithm 1.

First of all, both populations P1 and P2 are initialized by scout bees. After the solution construction of the bees in P1 they communicate with P2. If a solution φ^{eb} in P1 is better than the worst solution $\varphi^{worst'}$ in P2 regarding the fitness function F2 and if this solution is not already part of P2, then φ^{eb} substitutes solution $\varphi^{worst'}$. If the algorithm is already in the second stage, the solution construction by the bees of P2 follows in a similar way with a subsequent feedback to P1. If a solution $\varphi^{eb'}$ in P2 is better (regarding F1) than the worst solution φ^{worst} in P1 and is not part of P1, then φ^{worst} is substituted by $\varphi^{eb'}$.

Afterwards, the age of the solutions is checked. If a solution exceeds the maximum number of iterations without any improvement, this solution is substituted. An old solution in P1 is substituted by the solution of a scout bee, while a solution of P2 is substituted by the best solution regarding F2 from P1, which is not already contained in P2. Additionally, this only takes place if the solution possesses the best number of vehicles known so far. If such a solution does not exist, no substitution takes place in this iteration.

This procedure is repeated until a general stopping criterion, for instance a maximum number of iterations, is fulfilled.

Algorithm 1 Gerneral concept of BCiA

```
procedure BCiA()
 1: isFirstPop, isPhase1 ← TRUE
 2: InitializePopulations()                    /*as scout bees*/
 3: repeat
 4:   if stopping criterion is met for first phase then
 5:     isPhase1 ← FALSE                       /*shift to phase 2*/
 6:   end if
 7:   DoBeesSolutionConstruction(isFirstPop, P1)
 8:                /*communication with population P2*/
 9:   for all Empl. Bees eb in P1, ∄eb'∈P2: φ^eb' = φ^eb do
10:     φ^worst' ← worst solution φ^eb' in P2 regarding F2
11:     if F2(φ^eb) <F2(φ^worst') then
12:       φ^worst' ← φ^eb
13:     end if
14:   end for
15:   if not isPhase1 then
16:     DoBeesSolutionConstruction(not isFirstPop, P2)
17:                /*communication with population P1*/
18:     for all Empl. Bees eb' in P2, ∄eb∈P1: φ^eb = φ^eb' do
19:       φ^worst ← worst solution φ^eb in P1 regarding F1
20:       if F1(φ^eb') <F1(φ^worst) then
21:         φ^worst ← φ^eb'
22:       end if
23:     end for
24:   end if
25:         /*check age of solutions, replace if exceeded*/
26:   CheckSolutionAge(isFirstPop, P1)
27:   if not isPhase1 then
28:     CheckSolutionAge(not isFirstPop, P2)
29:   end if
30: until stopping criterion is met
```

5.2 Detailed Description

The function *DoBeesSolutionConstruction()* may be activated by P1 and P2 and handles the whole construction process of all solutions of the respective population within one iteration. First, the preferred paths of the follower bees are determined. This selection is done in a fitness-correlated and probabilistic way on basis of the solution quality of the employed bees. After the actual solution construction of a follower bee in function *ConstructSolution()*, a solution improvement is attempted. For population P1, this only refers to the attempt of vehicle reduction; additionally, for P2 a reduction of the total tour length is attempted. Both functions, which are described in detail below, only accept the new created solutions if an improvement regarding the respective fitness function has been achieved. After that, the employed bees are compared to the follower bees they have recruited. An employed bee is replaced by one of those follower bees if a follower bee is better than the employed bee, regarding the fitness function.

The solution construction of the scout bees follows in a similar way like the solution construction of the follower bees, except the fact that it is not given any path for orientation to the function *ConstructSolution()*. However, the attempt to reduce the number of vehicles and to reduce the total tour length in P2 still takes place. As soon as the solution construction of a scout bee is finished, it is checked whether one of the employed bees can be substituted by the newly created solution.

Algorithm 2 illustrates the procedure of the solution construction in Pseudocode.

Algorithm 2 Solution construction of the bees

procedure DoBeesSolutionConstruction($isFirstPop$, P)
1: /*solution construction of follower bees*/
2: CalculateSelectionProbabilities($isFirstPop$, P)
3: **for all** follower bees *fb* in P **do**
4: $fb.\varphi^{pref} \leftarrow$probabilistic choice of preferred path φ^{eb} in P
5: $\varphi^{fb} \leftarrow$ ConstructSolution($fb.\varphi^{pref}$)
6: DoTourReduction(φ^{fb}, *isFirstPop*)
7: DoLengthReduction(φ^{fb}, *isFirstPop*) /*only P2*/
8: **end for**
9: /*replace employed bee, if follower bee is better*/
10: **for all** follower bees *fb* in P **do**
11: **if** F1/F2(φ^{fb}) < F1/F2($\varphi^{eb} : \varphi^{eb} = fb.\varphi^{pref}$) **then**
12: $\varphi^{eb} \leftarrow \varphi^{fb}$
13: **end if**
14: **end for**
15: /*solution construction of scout bees*/
16: **for all** scout bees *sb* in P **do**
17: $\varphi^{sb} \leftarrow$ ConstructSolution(**null**)
18: DoTourReduction(φ^{sb}, *isFirstPop*)
19: DoLengthReduction(φ^{sb}, *isFirstPop*) /*only P2*/
20: $\varphi^{worst} \leftarrow$ worst solution φ^{eb} in P regarding F1/F2
21: **if** F1/F2(φ^{sb}) < F1/F2(φ^{worst}) **then**
22: $\varphi^{worst} \leftarrow \varphi^{sb}$
23: **end if**
24: **end for**

5.3 Application To The VRPTW

In the following section, it is shown how the explained general principle of the BCiA can be applied to the VRPTW. First of all, the function *ConstructSolution()*, in which the actual solution construction takes places, needs to be explained. The pseudo code of the function can be seen in algorithm 3.

Algorithm 3 Construction of a single solution

procedure ConstructSolution(φ^{pref})
1: **while** still unvisited customers **do**
2: $i \leftarrow \{0\}$
3: $\varphi^k \leftarrow \varphi^k + \{i\}$ /*new tour starting at the depot*/
4: $ct \leftarrow 0$, $load \leftarrow 0$
5: **while** $\mathcal{N}_i \neq \emptyset$ **do**
6: **if** $\varphi^{pref} \neq$ **null** and $ran \leftarrow [0,1) < q_0$ and $j^{pref} \in \mathcal{N}_i$ **then**
7: $j \leftarrow$ choice of j according to φ^{pref}
8: **if** $j = \{n+1\}$ **then**
9: **break** while-loop /*no reuse of vehicles*/
10: **end if**
11: **else**
12: $j \leftarrow$ probabilistic selection of $j \in \mathcal{N}_i \backslash \{0\}$ on basis of some heuristics value $\eta_{i,j}$
13: **end if**
14: $\varphi^k \leftarrow \varphi^k + \{j\}$ /*add next node to tour*/
15: $ct \leftarrow ct + t_{i,j} + w_j + s_j$
16: $load \leftarrow load + q_j$
17: $i \leftarrow j$ /*set bee to the current node*/
18: **end while**
19: $\varphi^k \leftarrow \varphi^k + \{n+1\}$ /*close tour by adding depot*/
20: **end while**

The BCiA can be classified as a metaheuristic of type I, because although the solutions are based on the formerly constructed solutions (preferred path) they are always created from the scratch.

The solution construction always starts with an empty route schedule in depot-node 0. The outer loop (lines 1-20) is repeated until there are no more unvisited customers left. Within this loop, a new tour is opened from the depot. Furthermore, the current time (ct) and the current load are set to zero (line 4). The inner loop (lines 5-18) is repeated until the neighborhood of the current node i is empty, i.e. no more customers can be added to the tour.

If the function was given a preferred path, so it is about the solution construction of a follower bee, a uniformly distributed random number ran in $[0,1)$ is created and compared to the parameter q_0 (line 6). This parameter controls the amount of exploitation of the algorithm in a similar way as it is done in ant algorithms based on *Ant Colony System* [10].[1] If $ran < q_0$ holds, the bee tries to select the preferred node j from the preferred path, provided it is still available, i.e. it lies in the neighborhood $\mathcal{N}_i$ (line 7). If the preferred node represents the depot, the inner loop aborts to avoid a reuse of the vehicles (lines 8-9). If the selection of the first customer of a new tour is concerned, one of the still possible first customers form one of the tours of the preferred path is selected randomly. This leads to a stronger diversification of the solution construction and avoids a situation, in which the constructed solutions are very similar to the preferred

[1] It needs to be noticed here that q_0 has to be adjusted to the size of the problem, i.e. that in case of a higher number of customers, a higher value of q_0 is needed to achieve an approximately equal deviation from the preferred path.

path at the beginning of the route schedule, while deviating from it strongly at the end.

If no preferred path is available, $ran \geq q_0$ holds or the preferred node j is not in $\mathcal{N}_i$, the next node j is determined probabilistically by means of a heuristics value $\eta_{i,j}$. Here, the way of calculating the urgency of visiting a certain node, suggested by GAMBARDELLA et al. for the MACS-VRPTW, was applied [13]. It must be noted that in this case the vehicles can only head for the depot if no further customers can be added to their tour. A premature return to the depot is only possible if it is determined by the preferred path.

If the customer j to be visited next is selected, it is added to the current tour, the time is adjusted, the vehicle's load is updated and the bee is set to the current node.

In the above section it was already mentioned that a solution improvement is intended after the solution construction. This concerns the reduction of the number of vehicle needed in every case and reduction of the total tour length for solutions of population P2. To reduce the number of tours, a method based on the procedure suggested by HOMBERGER [14] is applied. It is attempted to insert the customers of the smallest tour ι of a route schedule φ into other tours. If the smallest tour ι can be dissolved completely in this way, it is removed from the route schedule and the procedure is restarted with the now smallest tour ι'. If it is not possible to dissolve ι completely, the procedure is started with $\varphi' = \varphi \setminus \iota$ again and is continued until no complete resolution is possible any longer. Then however, there is a high potential to be able to insert the customers of the smallest tour ι into the smallest tour ι' of φ'.

The CROSS-operator by TAILLARD et al. was applied for minimizing the total tour length [26]. Always the best possible modification with the operator that leads to an improvement of the total tour length is carried out on the route schedule. Thus, the procedure is able to provide excellent solutions on the one hand but has a high complexity and is therefore poorly scalable on the other hand.

6. EVALUATION

To be able to assess the quality of the suggested algorithm, an evaluation was carried out on the basis of the standard benchmarks by SOLOMON [25]. These 56 problem instances contain 100 customers each, which differ in their positioning and the width of their assigned time windows. The results are shown in table 1 in an aggregated form and are compared to the most successful suggested procedures for the VRPTW. Apparently, the BCiA is able to produce solutions of a higher quality than it is possible with the above mentioned ant algorithms, which can also both be understood as metaheuristics of type I. Hence, the proposed BCiA represents the best metaheuristic of type I for the VRPTW. For each of the problem instances five solution runs were carried out with a maximum run time of 30 minutes each. The table shows the aggregated values for each of the problem classes. Besides the average and the best solution quality out of the five runs per problem instance, the best solution quality that has ever been reached is reported. With the BCiA it was even possible to reach a new best published value for the SOLOMON problem class R2.

The most important settings were: 5 employed bees, 25 follower bees, 5 scout bees, $q_0 = 0,95$, maximum number of iterations without improvement is 10, maximum run times in stage 1 and stage 2 are 900 seconds each.

7. CONCLUSIONS

This paper shows, using standard benchmark problems for the VRPTW, that bee algorithms are absolutely competitive compared to other metaheuristics. The proposed algorithm was particularly efficient for the smaller test entities of the SOLOMON benchmarks. The latest test runs for bigger benchmark instances show an increase of the gap between the BCiA and the most successful metaheuristics of type II. On the one hand, reasons for this might be found in the usage of the CROSS method, which seems to be too complex for bigger problem instances. On the other hand, advice may be found in literature (especially in HOMBERGER [14, p. 77]) that metaheuristics of type II are generally superior to those of type I regarding the expectable results. However, it must be stated that the presented approach achieves the best results compared to other metaheuristics of type I.

8. REFERENCES

[1] Aristotle. *The History of Animals*, volume III, books VII–X. Cambridge, Loeb Classical Library edition, 1991.

[2] A. Baykasoğlu, L. Özbakır, and P. Tapkan. Artificial bee colony algorithm and its application to generalized assignment problem. In F. Chan and M. Tiwari, editors, *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, pages 113–144. Vienna, 2007.

[3] R. Bent and P. van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4):515–530, 2004.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. New York, 1999.

[5] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[6] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

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

[8] C. S. Chong, A. I. Sivakumar, M. Y. H. Low, and K. L. Gay. A bee colony optimization algorithm to job shop scheduling. In L. F. Perrone et al., editors, *Proceedings of the 38th conference on winter simulation(WSC)*, pages 1954–1961, Monterey, 2006.

[9] C. S. Chong, A. I. Sivakumar, M. Y. H. Low, and K. L. Gay. Using a bee colony algorithm for neighborhood search in job shop scheduling problems. In *21st European Conference on Modelling and Simulation (ECMS)*, Prague, 2007.

[10] M. Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[11] M. Dorigo and T. Stützle. *Ant Colony Optimization*. Cambridge, 2004.

[12] I. Ellabib, P. H. Calamai, and O. A. Basir. Exchange strategies for multiple Ant Colony System. *Information Sciences*, 177(5):1248–1264, 2007.

Solomon Benchmarks		*BCiA*		*Gam99*		*Ell07**		*Pre07**		*Pis07**		*Ben04*	
		$f(\varphi)$	$g(\varphi)$	$f(\varphi)$	$g(\varphi)$	$f(\varphi)$	$g(\varphi)$	$f(\varphi)$	$g(\varphi)$	$f(\varphi)$	$g(\varphi)$	$f(\varphi)$	$g(\varphi)$
C1	avg.	**10**	**828.38**	**10**	**828.38**	-	-	**10**	**828.38**	**10**	**828.38**	**10**	**828.38**
	best	**10**	**828.38**	-	-	10	834.58	**10**	**828.38**	**10**	**828.38**	**10**	**828.38**
	best ever	**10**	**828.38**	**10**	**828.38**	10	834.58	**10**	**828.38**	**10**	**828.38**	**10**	**828.38**
R1	avg.	12.45	1197.34	12.38	1210.83	-	-	**11.92**	**1211.69**	12.03	1215.16	12.03	1213.21
	best	12.25	1201.1	-	-	12.33	1234.33	**11.92**	**1210.34**	11.92	1212.39	11.92	1213.54
	best ever	12.17	1203.87	12	1217.73	12.33	1234.33	**11.92**	**1210.34**	11.92	1212.39	11.92	1211.09
RC1	avg.	11.93	1372.54	11.92	1388.13	-	-	**11.5**	**1386.98**	11.6	1385.56	11.63	1380.06
	best	11.63	1378.89	-	-	11.88	1385.51	**11.5**	**1384.16**	11.5	1385.78	11.5	1384.22
	best ever	11.5	1388.7	11.63	1382.42	11.88	1385.51	**11.5**	**1384.16**	11.5	1385.78	11.5	1384.17
C2	avg.	**3**	**589.86**	3	591.85	-	-	**3**	**589.86**	**3**	**589.86**	**3**	**589.86**
	best	**3**	**589.86**	-	-	3	601.19	**3**	**589.86**	**3**	**589.86**	**3**	**589.86**
	best ever	**3**	**589.86**	**3**	**589.86**	3	601.19	**3**	**589.86**	**3**	**589.86**	**3**	**589.86**
R2	avg.	2.8	959.03	3	960.31	-	-	2.85	939.86	2.75	965.94	**2.73**	**985.36**
	best	2.73	960.17	-	-	3	1072.19	**2.73**	**955.74**	2.73	957.72	2.73	966.37
	best ever	**2.73**	**954.24**	2.73	967.75	3	1072.19	2.73	955.74	2.73	957.72	2.73	954.27
RC2	avg.	3.3	1131.91	3.33	1149.28	-	-	3.28	1115.68	**3.25**	**1135.46**	3.28	1156.39
	best	3.25	1131.15	-	-	3.38	1238.68	**3.25**	**1119.44**	3.25	1123.49	3.25	1141.24
	best ever	3.25	1121.27	3.25	1129.19	3.38	1238.68	**3.25**	**1119.44**	3.25	1123.49	3.25	1124.46
Runs		5		3		3		5		10		5	
CPU time		900/900 sec.		1800 sec.		1800 sec.		1800 sec.		avg. 146 sec.		6400 sec.	

Table 1: Evaluation results of the BCiA compared to other methods
The table shows the results of the BCiA as well as other methods for an application to the benchmark problems by Solomon [25]. *Gam99* stands for the method by Gambardella et al. [13], *Ell07* for the procedure by Ellabib et al. [12], *Pre07* for the algorithm by Prescott-Gagnon et al. [23], *Pis07* for the application by Pisinger et al. [22] and *Ben04* for the procedure by Bent/van Hentenryck [3]. Highlighted values represent the best solutions in the comparison. *As the best-ever-values for *Ell07*, *Pre07* and *Pis07* have not been published yet, those values have been taken from the best-values.

[13] L. M. Gambardella, É. Taillard, and G. Agazzi. MACS-VRPTW: A multiple Ant Colony System for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 63–76. London, 1999.

[14] J. Homberger. *Verteilt-parallele Metaheuristiken zur Tourenplanung*. Wiesbaden, 2000.

[15] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, 2005.

[16] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007.

[17] N. Lemmens. To bee or not to bee: A comparative study in swarm intelligence. Master's thesis, Maastricht University, Faculty of Humanities and Sciences, 2006. MICC-IKAT 06-12.

[18] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–227, 1981.

[19] P. Lučić. *Modeling Transportation Problems Using Concepts of Swarm Intelligence and Soft Computing*. PhD thesis, Virginia Polytechnic Institute and State University, 2002.

[20] S. Nakrani and C. Tovey. On honey bees and dynamic server allocation in internet hosting centers. *Adaptive Behavior*, 12(3–4):223–240, 2004.

[21] D. T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi. The Bees Algorithm – a novel tool for complex optimisation problems. In D. T. Pham, E. E. Eldukhri, and A. J. Soroka, editors, *Proceedings of the 2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS)*, pages 454–459, Amsterdam, 2006.

[22] D. Pisinger and S. Røpke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34(8):2403–2435, 2007.

[23] E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. Technical Report G-2007-67, GERAD and École Polytechnique de Montréal, 2007.

[24] M. W. P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4:285–305, 1985.

[25] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Opererations Research*, 35(2):254–265, 1987.

[26] É. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31:170–186, 1997.

[27] D. Teodorović and M. Dell'Orco. Bee colony optimization – a cooperative learning approach to complex transportation problems. In A. Jaszkiewicz et al., editors, *Advanced OR and AI Methods in Transportation*, pages 51–60, Poznan, 2005.

[28] K. von Frisch. *The Dance Language and Orientation of Bees*. 1967.

[29] H. F. Wedde, M. Farooq, and Y. Zhang. Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In M. Dorigo et al., editors, *Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS)*, pages 83–94, Berlin, 2004.