Bee Colony Optimization Overview

Dušan Teodorović [1], Tatjana Davidović[2] and Milica Šelmić [1]

[1] University of Belgrade, Faculty of Transport and Traffic Engineering,
Vojvode Stepe 305, 11000 Belgrade, Serbia
{dusan,m.selmic}@sf.bg.ac.rs
[2] Mathematical Institute, Serbian Academy of Sciences and Arts,
Kneza Mihaila 36, P.O.Box 367, 11000 Belgrade, Serbia
tanjad@mi.sanu.ac.rs

Abstract. The Bee Colony Optimization (BCO) meta-heuristic belongs to the class of Nature-Inspired Algorithms. This technique uses an analogy between the way in which bees in nature search for a food, and the way in which optimization algorithms search for an optimum in combinatorial optimization problems. Artificial bees represent agents, which collaboratively solve complex combinatorial optimization problem. The chapter presents a description of the algorithm, classification and analysis of the results achieved using Bee Colony Optimization (BCO) to model complex engineering and management processes.

1 Introduction

The Bee Colony Optimization (BCO) meta-heuristic [29, 30, 31, 32] belongs to the class of *Nature-Inspired Algorithms*. These algorithms are inspired by various biological and natural processes. Natural systems have become important sources of ideas and models for development of various artificial systems. The popularity of the Nature-Inspired Algorithms is mainly caused by the capability of biological systems to successfully adjust to continually varying environment. Neural networks, evolutionary computation, ant colony optimization, particle swarm optimization, artificial immune systems, and bacteria foraging algorithm are some of the algorithms and concepts that were inspired by nature.

Individuals in various biological systems are engaged in cooperation, collaboration, information exchange, and/or conflicts. In many cases, individuals, that are autonomous in their decision-making, work together with other individuals in order to achieve specific objective. Natural phenomena lecture us that simple individual organisms can create systems able to perform highly complex tasks by interacting with each other.

Few algorithms inspired by bees' behavior appeared during the last decade (Bee System, BCO algorithm, ABC algorithm, MBO, Bees Algorithm, HBMO algorithm, BeeHive, VBA algorithm) [1, 2, 4, 11, 19, 21, 23, 24, 25, 26, 28, 35, 36, 37, 38, 40, 41, 42, 43, 55, 56, 57, 58, 59, 60]. Yonezava and Kikuchi [60] analyzed collective intelligence based on bees' behavior. Sato and Hagiwara [43] proposed modified genetic algorithm named Bee System. In essence, this algorithm belongs to the class of genetic algorithms. Abbas [1] developed *MBO* model that is based on the marriage process in honeybees. *BeeHive* [55, 56, 57], *Artificial Bee Colony* (*ABC*) *algorithm* [23, 24, 25] and *Bees Algorithm* [36, 37, 38] are based on foraging behavior in honeybees but all of them use different concepts for algorithm development. An excellent survey of the Bees' behavior inspired algorithms could be found in Baykasoglu et al. [3].

The BCO meta-heuristic [29, 30, 31, 32] has been proposed quite recently by Lučić and Teodorović. The BCO is inspired by foraging behavior in honeybees. (Lučić and Teodorović used the term "Bee System" in their first paper). The basic plan behind the BCO is to build the multi agent system (colony of artificial bees) able to efficiently solve hard combinatorial optimization problems. The artificial bee colony behaves partially similar, and partially in a different way from bee colonies in nature.

The BCO meta-heuristic has been recently used as a toll for solving large and complex real-world problems. It has been shown that the BCO poses an ability to find high quality solutions of difficult combinatorial problems within a reasonable amount of computer time. The BCO is a stochastic, random-search technique. This technique uses an analogy between the way in which bees in nature search for a food, and the way in which optimization algorithms search for a optimum of (given) combinatorial optimization problems. The basic idea behind the BCO is to build the multi agent system (colony of artificial bees) able to effectively solve difficult combinatorial optimization problems. Artificial bees investigate through the search space looking for the feasible solutions. In order to find better and better solutions, autonomous artificial bees collaborate and exchange information. Using collective knowledge and sharing information among themselves, artificial bees concentrate on more promising areas, and slowly abandon solutions from the less promising areas. Step by step, artificial bees collectively generate and/or improve their solutions. The BCO search is running in iterations until some predefined stopping criteria is satisfied.

The BCO works in a self-organized and decentralized way and therefore represents a good basis for parallelization. It also poses an ability to keep away from becoming trapped in local minima.

This chapter presents a description of the BCO and some of its modifications, as well as the classification and analysis of the results achieved using BCO to model complex engineering and management processes. We initially portray the behavior of bees' in nature, and then we describe a general Bee Colony Optimization algorithm. Afterwards, we present some modifications of BCO that allow its application to some non-standard combinatorial optimization problems. Later on, we describe BCO applications in different engineering and management problems. The BCO has been successfully applied to various engineering and management problems by Teodorović and coauthors [16, 17, 18, 20, 33, 44, 45, 46, 47, 48, 49, 50, 51]. The BCO has been applied in the cases of the Traveling Salesman Problem [29, 30, 31], the Ride-Matching Problem [48, 49], the Routing and Wavelength Assignment (RWA) in All-Optical Networks [33], the *p*-median problem [51], static scheduling of independent tasks on homogeneous multiprocessor systems [17, 18], and traffic sensors locations problem on highways [20, 44].

2 Biological Background

Swarm behavior (fish schools, flocks of birds, and herds of land animals) is based on the *biological needs* of individuals to stay together. When staying together, individuals have a higher probability to stay alive, since predator usually attacks only one individual. Flocks of birds, herds of animals, and fish schools are characterized by collective movement. Herds of animals react at once to changes in the course and speed of their neighbors.

Colonies of various social insects (bees, wasps, ants, termites) are also characterized by swarm behavior. Swarm behavior is primarily characterized by autonomy, distributed functioning and self-organizing. The communication systems between individual insects contribute to the pattern called the ‘‘collective intelligence” of the social insect colonies. The term ‘‘Swarm Intelligence”, that denotes this ‘‘collective intelligence”, has been introduced in [5, 6, 7, 8].

Swarm Intelligence [8] is the branch of Artificial Intelligence. Swarm Intelligence is based on investigation of actions of individuals in different decentralized systems. These decentralized systems (Multi Agent Systems) are composed of physical individuals (robots, for example) or “virtual” (artificial) ones that communicate among themselves, cooperate, collaborate, exchange information and knowledge and perform some tasks in their environment. When designing Swarm Intelligence models, researchers use some principles of the natural swarm intelligence. The development of artificial systems usually does not involve the entire imitation of natural systems, but explores them while searching for ideas and models.

As we already mentioned, the BCO is inspired by foraging behavior of honeybees. Bees in nature look for a food by exploring the fields in the neighborhood of their hive. They collect and accumulate food for later use by other bees. Typically, in the initial step, some scouts search the region. Completing the search, scout bees will return to the hive and inform their hive-mates about the locations, quantity and quality of available food sources in the areas they have examined. In case they have discovered nectar in the previously investigated locations, scout bees will dance in the so-called “dance floor area” of the hive, in an attempt to “advertise” food locations and encourage the remaining members of the colony to follow their lead. The information about the food quantity is presented using a ritual called a “waggle dance”. If a bee decides to leave the hive to collect nectar, it will follow one of the dancing scout bees to the previously discovered patch of flowers. Upon arrival, the foraging bee takes a load of nectar and returns to the hive relinquishing the nectar to a food storer bee. Several scenarios are then possible for a foraging bee: (1) it can abandon the food location and return to its role of an uncommitted follower; (2) it can continue with the foraging behavior at the discovered nectar source, without recruiting the rest of the colony; (3) it can recruit its hive-mates with the dance ritual before the return to the food location. The bee opts for one of the above alternatives with a certain probability. The described process continues repeatedly, while the bees at a hive accumulate nectar and explore new areas with potential food sources. As several bees may be attempting to recruit their hive-mates at the dance floor area at the same time, it is unclear how a bee resting at a hive decides which dancing bee to follow, although it has been considered that “the recruitment among bees is always a function of the quality of the food source” [10].

3 Bee Colony Optimization (BCO) Algorithm

The BCO belongs to the class of population-based algorithms. It has been proposed for the first time in [29, 30, 31] and was evolving through later applications. The early versions of the algorithm were imitating the behavior of the bees in the nature to a larger extent. These versions were characterized by the *scout* bees, an important role of the hive location, and recruiting process that is more like the natural one than it is the case in the current version of the algorithm. In this section we describe in details the current version and we will point out the differences while presenting the concrete applications.

Population of agents (artificial bees) consisting of *B* bees collaboratively searches for the optimal solution. Every artificial bee generates one solution to the problem. There are two alternating phases (*forward pass* and *backward pass*) constituting single step in the BCO algorithm. In each forward pass, every artificial bee explores the search space. It applies a predefined number of moves, which construct and/or improve the solution, yielding to a new solution. For example, let bees Bee 1, Bee 2, …, Bee *B* participate in the decision-making process on *n* entities. At each forward pass bees are supposed to select one entity. The possible situation after third forward pass is illustrated on Fig.1.

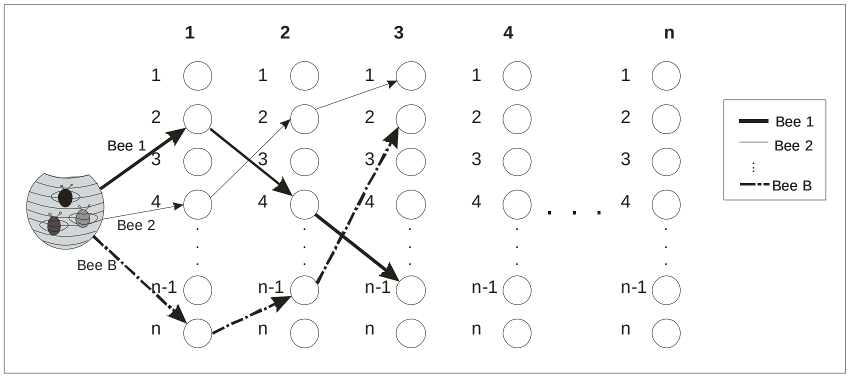

Fig. 1. An illustration of the third forward pass

Having obtained new partial solutions, the bees go again to the hive and start the second phase, the so-called backward pass. In the backward pass, all artificial bees share information about the quality of their solutions. In nature, bees would return to the hive, perform a dancing ritual, which would inform other bees about the amount of food they have discovered, and the proximity of the patch to the hive. In the search algorithm, the bees announce the quality of the solution, i.e. the value of objective function is computed. Having all solutions evaluated, every bee decides with a certain probability whether it will stay loyal to its solution or not. The bees with better solutions have more chances to keep and advertise their solutions. On the contrary to the bees in nature, artificial bees that are *loyal* to their partial solutions are at the same time *recruiters*, i.e. their solutions would be considered by other bees. Once the solution is abandoned by a bee it becomes *uncommitted* and has to select one of the advertised solutions. This decision is taken with a probability too, so that better advertised solutions have bigger opportunity to be chosen for further exploration. In such a way, within each backward pass all bees are divided into two groups (*R* recruiters, and remaining *B*-*R* uncommitted bees) as it is shown on Fig. 2. Values for *R* and *B*-*R* are changing from one backward pass to another one.

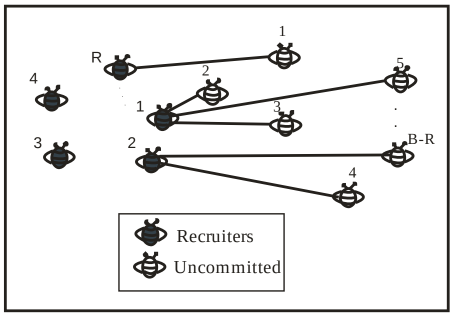

Fig. 2. Recruiting of uncommitted followers

After comparing all generated partial solutions, Bee 2, from the previous example decided to abandon already generated solution and to join Bee *B* (see Fig.3).

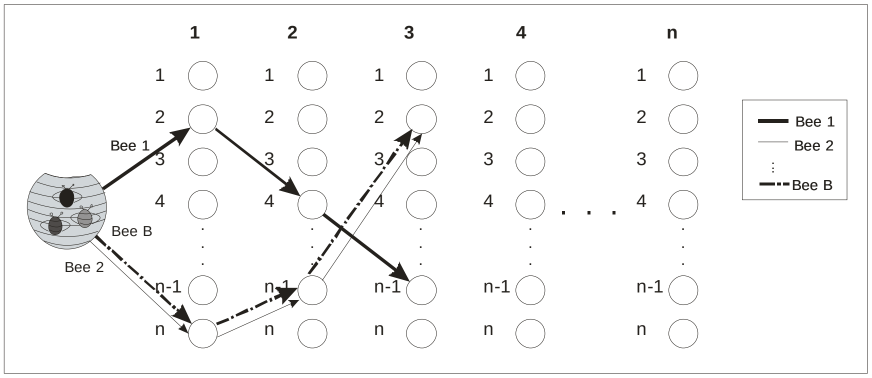

Fig. 3. The possible result of a recruiting process within third backward pass

Bee 2 and Bee *B* "fly together" along the path already generated by the Bee *B*. In practice, this means that partial solution generated by Bee *B* is associated (copied) to Bee 2 also. When they reach the end of that common path, they are free to make an individual decision about the next constructive step to be made. The Bee 1 will keep already generated partial solution without being chosen by any of the uncommitted hive-mates, and therefore, it will perform new constructive step independently. The described situation is illustrated on Fig.4.

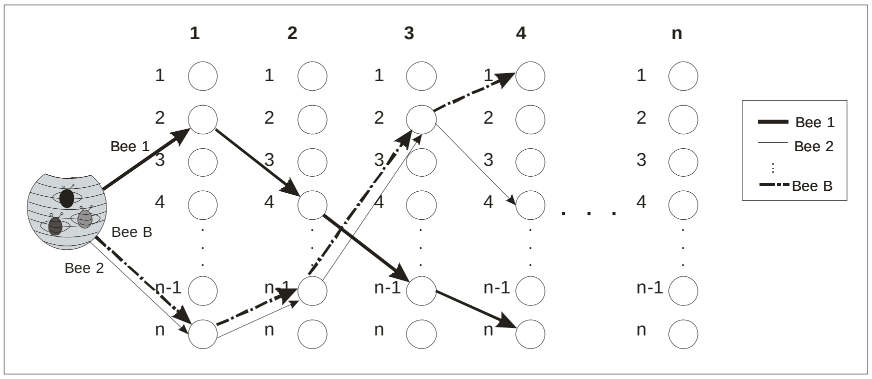

Fig. 4. An example of partial solutions after fourth forward

The two phases of the search algorithm, forward and backward pass, are alternating in order to generate all required feasible solutions (one for each bee). When all solutions are completed the best one is determined, it is used to update global best solution and an iteration of the BCO is accomplished. At this point all *B* solutions are deleted, and the new iteration could start. The BCO runs iteration by iteration until a stopping condition is met. The possible stopping conditions could be, for example, the maximum total number of forward/backward passes, the maximum total number of forward/backward passes without the improvement of the objective function, maximum allowed CPU time, etc. At the end, the best found solution (the so called global best) is reported as the final one.

The algorithm parameters whose values need to be set prior the algorithm execution are as follows:

B - The number of bees in the hive;
NC - The number of constructive moves during one forward pass.

At the beginning of the search, all the bees are in the hive. According to the main idea in the current version of the BCO algorithm, the hive is an artificial object, without precise location and does not influence the algorithm execution. It is used only to denote the synchronization points at which bees are exchanging information about the current state of the search. The pseudocode of the BCO algorithm could be described in the following way:

```
1.  Initialization: an empty solution is assigned to every bee;
2.  For every bee: // the forward pass
        i. Set $k = 1$;  //counter for constructive moves in the forward pass;
        ii. Evaluate all possible constructive moves;
        iii. According to evaluation, choose one move using the roulette wheel;
        iv. $k = k + 1$; If $k \leq NC$ **Goto** step ii.
3.  All bees are back to the hive; // backward pass starts;
4.  Evaluate (partial) objective function value for each bee;
5.  Every bee decides randomly whether to continue its own exploration and become a recruiter, or to become a follower;
6.  For every follower, choose a new solution from recruiters by the roulette
    wheel;
7.  If solutions are not completed **Goto** step 2;
8.  Evaluate all solutions and find the best one;
9.  If the stopping condition is not met **Goto** step 2;
10. Output the best solution found.
```

3.1 Constructive and Improving Alternatives of the BCO Algorithms

Until now, the BCO algorithms in the literature have been constructive. The BCO starts from scratch and, for each bee, constructs a solution step by step applying some stochastic problem specific heuristics. Randomness induced by these stochastic construction processes assures diversity of the search. Within each iteration *B* solutions are generated and the best of them is used for updating the current global best solution. Next iteration then results in *B* new solutions among which we search for the new global best one.

The BCO could also work as an improving algorithm. In this case, the analyst would start from a complete solution. The complete solution could be generated randomly or by some heuristics. By perturbing that solution, artificial bees would try to improve it. Todorović et al. in [53] developed a bee colony approach for the nurse rostering problem. Their approach is the first one that allows both constructive and improving steps to be applied and combined together. The developed algorithm was designed to be a combination of constructive and local search phases. In the constructive phase, unscheduled shifts are assigned to available nurses. A local search move could be applied to both partial and complete rosters. Its role was to modify a roster either by swapping assignments of nurses, or by reassigning a shift to another nurse.

The proposed approach also incorporated a novel intelligent discarding of portions of large neighborhoods for which it is predicted that they will not lead to the improvement of the objective function. The performance of the algorithm was evaluated on real world data from hospitals in Belgium.

The idea of improving alternatives could be developed in many different ways, and this approach certainly may be very useful for solving difficult combinatorial optimization problems.

3.2 The Artificial Bees and Fuzzy Logic

In most of the models it is assumed that problem data (costs, capacity, distance, duration etc.) are *deterministic* quantities known in advance. On the other hand, the travel time between two nodes in a network, for example, involves an uncertainty due to traffic conditions, type of driving, weather conditions, choice of streets, and so on. Our subjective feeling regarding travel time is often not very precise. For example, the estimate is made that it takes “approximately 30 minutes” to go from one node to another. No one will claim that it takes 27 minutes when subjectively estimating travel time. Estimating travel time in this way is not the result of objective measurements but is a subjective estimation that differs among drivers. Travel time has often been treated as a *random variable*, and this treatment required travel time measurements and the establishment of a certain probability density function. However, dispatchers-decision makers most often make a subjective estimate of travel time based on their experience and intuition, expressing the estimated travel time as “short,” “long,” “about 20 minutes,” and so on. Travel time between two nodes in a network can be treated as *fuzzy number*. (Most sets in reality have no sharp line between the elements in the set and those outside the set. The simplest examples of fuzzy sets are classes of elements characterized by adjectives: *big, small, fast, old,* etc. With fuzzy sets the membership function is associated and it takes continuous values from the closed interval [0,l]. Fuzzy set **A** is defined as a set of ordered pairs **A** = $\{x, \mu_A(x)\}$ where $\mu_A(x)$ indicates the grade of membership of element *x* in set **A** [61]. For example, if *x* is the travel time between two nodes, then *short* could be considered as a particular value of the fuzzy variable *travel time.* To each *x* a number $\mu_A(x) \in [0,1]$ is assigned. This number shows the extent to which *x* is considered to be short.).

The fuzzy set of subjectively estimated travel time between nodes *i* and *j* is denoted by **T**. In order to simplify the arithmetic operations, the travel time **T** is assumed to be a triangular fuzzy number. Triangular fuzzy number **T** is expressed as:

$$\mathbf{T} = (t_1, t_2, t_3) \tag{1}$$

where t_1, t_2, and t_3 are the lower boundary, the value that corresponds to the highest grade of membership, and the upper boundary of fuzzy number **T**, respectively [27].

The BCO application on models with fuzzy logic is almost the same as application on deterministic models. Main differences are in the part when:

- bees’ partial solutions are compared;
- different component attractiveness is calculated.

In the first case, Kaufmann and Gupta's method [27] can be used to compare fuzzy numbers. In the second case, the approximate reasoning algorithm for calculating the solution component attractiveness could be applied. This algorithm is usually composed from the rules of the following type (Fig. 5):

If the attributes of the solution component are VERY GOOD
Then the considered solution component is VERY ATTRACTIVE

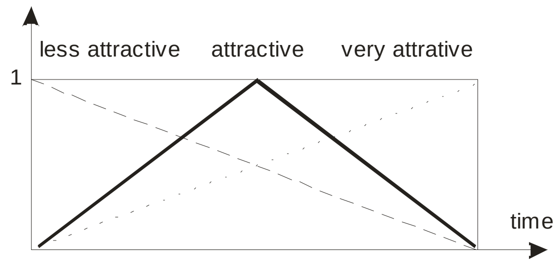

Fig. 5. Fuzzy sets describing attractiveness

The approximate reasoning based on Fuzzy Logic has been used in [32] to model uncertain demands in nodes when solving vehicle routing problem and in [48, 49] to model some uncertain quantities for solving Ride-Matching problem.

3.3 Parallelization of Bee Colony Optimization

The BCO algorithm created as a multi agent system provides a good basis for the parallelization on different levels. It seems to have significant amount of inherent parallelism. Therefore, studying the potential strategies for its parallelization, represent fruitful research field. As of the authors' knowledge, the only paper proposing strategies for parallelization of the BCO is [16]. In this subsection we will describe those and discuss some other potential parallelization strategies of BCO. First we give short description of specific points in parallelization of meta-heuristics (stochastic search algorithms for combinatorial optimization), an overview of parallel meta-heuristic classification and taxonomy, and then we describe parallelization of BCO based on the synchronous strategy.

The main goal of parallelization of any algorithm is to speedup the computations needed to solve a particular problem by engaging several processors and divide the total amount of work between them. For stochastic algorithms this goal may be defined in one of the following two ways: 1) accelerate the search for the same quality solution or 2) improve the solution quality by allowing more processors to run the same amount of (CPU or wall-clock) time as the single one does. When meta-heuristics are in consideration, the combination of gains may be obtained: parallel execution can enable efficient search of different regions of the solution space yielding to the improvement of the final solution quality within smaller amount of execution time.

A significant amount of work has already been done on parallelization of meta-heuristics. The approach can be twofold. The theoretical aspects of parallelization could be considered, and practical applications of parallel meta-heuristics to different optimization problems proposed. Different parallelization strategies had been proposed in the recent literature dealing with various meta-heuristic methods [13, 22, 39]. The survey papers [12, 54] summarize these works and propose adequate taxonomy.

One of the first classifications of parallelization strategies was proposed in [54]. It is based on the control of the search process and results in two main groups of parallelization strategies: *single walk* and *multiple walks* parallelism. *Single walk* parallelization assumes that the unique search trajectory is generated and only required computations are performed in parallel. It is connected to the fine granularity of tasks to be executed in parallel and usually, it is devoted to speedup the execution without affecting the final solution quality.

Multiple walks parallelization strategy involves different search trajectories explored by different processors. It assumes medium to coarse granulation of tasks and they could be executed independently or in cooperation. The simplest example of multiple walks parallel search is independent run. It is the parallel simulation of the multistart execution and it does not involve information exchange during the search. Cooperative execution assumes data exchange during the search which affects the search trajectory on each processor.

To refine the classification of parallelization strategies, one has to consider communication aspects (synchronous or asynchronous) and search parameters (same or different initial point and/or same or different search strategies). The resulting classification is described in details in [12].

As we already mentioned, the BCO algorithm created as a multi agent system, provides a good basis for the parallelization on different levels. High level parallelization assumes coarse granulation of tasks and can be applied to iterations of BCO. Smaller parts of BCO algorithms (forward and backward passes within a single iteration) also contain a lot of independent executions and are suitable for low level parallelization. In [16] the authors considered both strategies in a synchronous way.

High level parallelization in its simplest form represents the independent execution of BCO on different processors. It could be obtained by the division of stopping criteria among processors. For example, if the stopping criteria is allowed CPU time (given as a *runtime* value in seconds), the BCO could run in parallel on q processors for *runtime*/q seconds. Similar rule can be introduced in the case when stopping criteria is allowed number of iterations. In both cases each processor performs independently sequential variant of BCO, but with reduced value of the stopping criteria. This variant of parallelized BCO was named *distributed BCO* (DBCO). Other way to implement coarse grained parallelization strategy could be the following: Instead of the stopping criterion the number of bees could be divided. Namely, if sequential execution uses B bees for the search, parallel variant executing on q processors would be using B/q bees only. This way results in sequential BCO on each processor, but with reduced number of bees. This variant is also distributed BCO, but was referred to as BBCO since the bees were distributed among processors.

Independent runs on different processors allow also changes of search parameters and therefore, this parallelization strategy belongs to the multiple walks group. Name-

Namely, once the stopping criteria is reduced, different values could also be assigned to the number of bees *B* and/or to the number *NC* of constructive moves within a single forward pass for each BCO executing on different processors. Similarly, for BBCO the number of bees does not have to be equally distributed among processors: the variant in which for each processor different number of bees (between 1 and *B*) is assigned with the same value for *NC* would provide different searches on different processors and therefore would represent the multiple walks parallelization strategy.

The implementation of low level parallelization strategy in [16] was named FBCO and it was based on the following facts. Each artificial bee acts as an individual agent during the forward pass when partial solutions are generated. The generation of partial solution is independent from the rest of the computations. This enables the fine level parallelization (the one from single walk group). Within the concrete implementation, in [16] the following scenario appeared: forward pass is executed independently on different processors while backward pass requires tight coordination between processors. For the corresponding computations within backward pass it is necessary to have the information about all generated partial solutions. Nevertheless, those computations could be done either sequentially by a single processor (master), or spread among all processors and accompanied by required communication. This communication is known to be the main bottleneck of parallel execution if distributed memory multiprocessor system is used. In that case, it is important to reduce both the amount of data and the number of transfers (messages).

The implementation of FBCO required the authors to define the relation between the number of processors *q* and the number of bees *B*. Namely, each processor was responsible for *B*/*q* bees, and therefore, these two numbers should be divisible.

There are some other possible parallelization strategies that could be applied to BCO and that belong to the classification proposed in [12]. For example, the bees could be allowed to perform several forward-backward passes before initiating communication between processors. During that period, loyalty could be determined only with respect to the bees from the same processor. Once communication is initiated, bees from different processors should have higher probability to be chosen by an uncommitted follower. The other way could be to develop an architecture dependent parallelization strategy. For example, within a ring topology, a processor communicates only with its neighbors. Therefore, the partial solutions are sent only to the next processor, and received only from the previous one in the ring. Again, the decisions about the loyalty would be made based on incomplete information. The above mentioned parallelization strategies already introduce asynchronous concepts in BCO, but it could be even more developed within potential future research.

The well known performance measures for parallel programs are *speed-up* S_q and *efficiency* E_q [9, 42]. They are defined as follows:

$$S_q = \frac{T_{seq}^{best}}{T_q}, \quad E_q = \frac{S_q}{q} = \frac{T_{seq}^{best}}{qT_q} \qquad (2)$$

Here, T_{seq}^{best} denotes the execution time of best known sequential algorithm on a single processor, while T_q represents the execution time of the parallel algorithm on q processors.

When meta-heuristics are under consideration, the performance of parallelization strategy is influenced also by the quality of final solution. Namely, meta-heuristics represent stochastic search procedures (and BCO is not an exception) which may not result with a same solution even in repeated sequential executions. On the other hand, parallelization may assure the extension of the search space which could yield to both improvement or degradation of the final solution quality. Therefore, the quality of final solution should also be considered as a parameter of parallelization strategy performance.

4 BCO Applications

4.1 Solving the Traveling Salesman Problem by BCO

Lučić and Teodorović [29, 30, 31] tested the Bee Colony Optimization approach in the case of Traveling Salesman Problem (TSP). The well known Traveling Salesmen Problem is defined in the following way: Given n nodes, find the shortest itinerary that starts in a specific node, goes through all other nodes exactly once and finishes in the starting node.

When solving the TSP problem the authors were also developing the BCO algorithm and it had more similarities with the behavior of bees in the nature, than the recent version of algorithm. The main difference between these two versions is in the fact that hive had an important role in the previous one. The hive had specified location that could also be changed during the search process. The other difference is that not all the bees are engaged at the beginning of the search process. The *scout* bees start the search, and at each stage new bees join it by recruiting process.

In [29, 30, 31] the authors locate hive at random node and decompose the TSP problem into stages. At each stage (corresponding to the forward pass of BCO), a bee chooses the new nodes to be added to the partial Traveling Salesman tour created so far. This selection was performed in random manner with certain probabilities. Lučić and Teodorović [29, 30, 31] proposed Logit-based model for calculating the probability of choosing next node to be visited. Logit model is one of the most successful and widely accepted discrete choice model [34]. When calculating this probability, the proposed model took into account the distance between current node (and/or hive) and node-candidate to be visited, the total number of performed iterations in a search process, as well as the total number of bees that visited considered link in the past. The proposed model was represented by the complex and complicated formulae, and was not used in subsequent research by other researchers.

During the backward pass each bee decided whether to abandon the generated partial solution (i.e. return to its role of an uncommitted follower) or keep it (i.e. dance to recruit the hive-mates that would follow it at the beginning of the next forward pass). There existed certain probabilities for these two choices, where bees with higher objective function value had greater chance to continue their own exploration. Each follower bee had chosen a new solution from one of the recruiters by the roulette wheel, where better solutions had higher probability of being chosen for exploration. After the selection had been made, bees expanded previously generated partial solutions by a predefined number of nodes during the next forward pass, followed by the second backward pass and return to the hive. Once in the hive, bees took part in a decision making process again, thus repeating the described process. These steps were repeated until complete solutions have been generated (for each bee the whole TSP tour was discovered). The authors tried to improve the solutions obtained by the bees in current iteration by applying different tour improvement algorithms based on *k*-opt procedure. Among all generated solutions, the best one was determined and used to update the global best. This represented the end of single iteration and the next one started after the hive relocation.

The authors explored the effectiveness of the BCO on a large number of numerical examples. Here we present the results for benchmark problems that were taken from the following Internet address:

http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/tsp/.

All tests were run on an IBM compatible PC with PIII processor (533MHz). The obtained results are given in Table 1.

Table 1. TSP benchmark problems: The results obtained by the BCO algorithm

Problem name	No. of nodes	Optimal value	BCO	Relative error (%)	CPU (sec)
Eil51	51	428.87	428.87	0.00	29
Berlin52	52	7544.37	7544.37	0.00	0
St70	70	677.11	677.11	0.00	7
Pr76	76	108159.00	108159.00	0.00	2
Kroa100	100	21285.40	21285.40	0.00	10
Eil101	101	640.21	640.21	0.00	61
Tsp225	225	3859.00	3899.90	1.06	11651
A280	280	2586.77	2608.33	0.83	6270
Pcb442	442	50783.55	51366.04	1.15	4384
Pr1002	1002	259066.60	267340.70	3.19	28101

Results given in the Table 1 show that the BCO proposed in [29, 30, 31] produced results of a very high quality. The BCO was capable to get the objective function values equal or very close to the optimal ones. The CPU times necessary to discover the best solutions by the BCO were very low (in 2001). In other words, the BCO was able to produce “very good” solutions in a “reasonable amount” of computer time.

4.2 Solving the Ride-Matching Problem by BCO

In a lot of countries urban road networks are highly congested. The negative consequences of traffic congestion are enlarged travel times, bigger number of stops, unanticipated delays, greater travel cost, inconvenience to drivers and passengers, increased air pollution, noise level and number of traffic accidents. Growing traffic network capacities by building more roads is extremely costly as well as environmentally devastating. Efficient usage of the existing supply is essential in order to sustain the growing travel demand. Researchers, planners, and transportation professionals have developed various Travel Demand Management (TDM) techniques. One of the widely used Travel Demand Management (TDM) techniques is ridesharing. Within this concept, two or more persons share vehicle when traveling from their origins to the destinations. The operator of the system must posses the following information regarding trips planned for the next week: (a) Vehicle capacity (2, 3, or 4 persons); (b) Days in the week when person is ready to participate in ride-sharing; (c) Trip origin for every day in a week; (d) Trip destination for every day in a week; (e) Desired departure and/or arrival time for every day in a week.

The ride-matching problem considered by Teodorović and Dell'Orco in [48, 49] could be defined in the following way: Make routing and scheduling of the vehicles and passengers for the whole week in such a way to minimize the total distance traveled by all participants. In [48, 49] the authors developed BCO based model for the ride-matching problem. They started their choice model from the assumption that the quantities perceived by bees are 'fuzzy'. They created artificial bees that use approximate reasoning and rules of fuzzy logic in their communication and acting. The main advantage of using the approximate reasoning algorithm for calculating the solution component attractiveness was that it made possible to calculate solution component attractiveness even if some of the input data were only approximately known. If f_i denotes the attractiveness value of solution component i, the probability p_i for solution component i to be added to the partial solution was equal to the ratio of f_i and the sum of all considered solution component attractiveness values:

$$p_i = \frac{f_i}{\sum_j f_j} \tag{3}$$

In order to choose the next solution component to be added to the partial solution, artificial bees use a proportional selection known as 'roulette wheel selection.' (The sections of roulette are in proportion to probabilities p_i). In addition to the 'roulette wheel selection,' several other ways of selection could be used. When adding the solution component to the current partial solution during the forward pass, a specific bee perceives a specific solution component as 'less attractive', 'attractive', or 'very attractive'. Artificial bee can perceive a specific attribute as 'short', 'medium' or 'long'; 'cheap', 'medium', or 'expensive'; etc. The authors developed the approximate reasoning algorithm for calculating the solution component attractiveness.

In order to describe bee's partial solutions comparison mechanism, the authors introduced the concept of partial solution badness. The partial solution badness was calculated in the following way:

$$L_k = \frac{L^{(k)} - L_{\min}}{L_{\max} - L_{\min}} \tag{4}$$

where
L_k – represents the badness of the partial solution discovered by the *k*th bee;
$L^{(k)}$ – is the objective function values of the partial solution discovered by the *k*th bee;
$L_{\min}$ and $L_{\max}$ denote the objective function value of the best and worst partial solution discovered from the beginning of the search process.

The approximate reasoning algorithm to determine bee's loyalty to its partial solution contained the rules of the following type:

If the discovered partial solution is BAD
Then loyalty is LOW

Bees use approximate reasoning, and compare their discovered partial solutions with the best, and the worst discovered partial solution from the beginning of the search process. In this way, 'historical facts' discovered by the all members of the bee colony have significant influence on the future search directions.

Based on the quality of its solution each bee decided with certain probability weather to stay loyal or became an uncommitted follower. Every partial solution (partial path) that was being advertised in the dance area had two main attributes: (a) the objective function value; and (b) the number of bees that were advertising the partial solution (partial path). The number of bees advertising the partial solution was a good indicator of a bees' collective knowledge. It showed how a bee colony perceives specific partial solutions. The authors used the approximate reasoning algorithm to determine the advertised partial solution attractiveness. It consisted of the rules of the following type:

If the length of the advertised path is SHORT
and the number of bees advertising the path is SMALL
Then the advertised partial solution attractiveness is MEDIUM

The approximate reasoning algorithm was used to calculate the number of shifting bees with the rules of the following type:

If bees' loyalty to path p_i is LOW
and path p_j 's attractiveness is HIGH
Then the number of shifting bees from path p_i to path p_j is HIGH

In this way, the number of bees flying along a specific path is changed before beginning of the new forward pass. Using collective knowledge and sharing information among themselves, bees concentrate on more promising search paths, and slowly abandon less promising paths.

Proposed model was tested in the case of ride-sharing demand from Trani, a small city in the south-east of Italy, to Bari, the regional capital of Puglia. The authors collected data regarding 97 travelers demanding ride sharing, and assumed, for sake of simplicity, that capacity is four passengers for all their cars. Fig.6 shows changes in the best-discovered objective function values through the iterations.

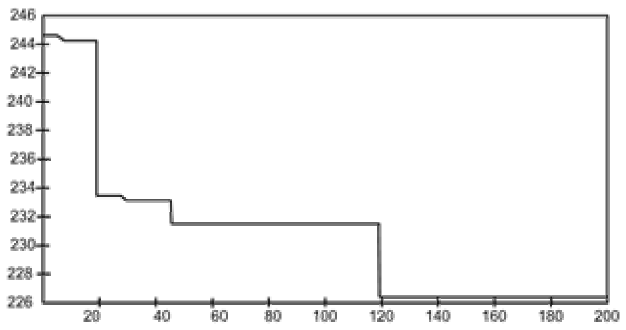

Fig. 6. Changes in the best-discovered objective function values through the iterations

4.3 Routing and Wavelength Assignment in All-Optical Networks Based by BCO

The Routing and Wavelength Assignment (RWA) in All-Optical Networks is the well known optimization problem in telecommunication. Every pair of nodes in optical networks is characterized by a number of requested connections. The total number of established connections in the network depends on the routing and wavelength assignment procedure. The RWA problem could be described in the following way: Assign a path through the network and a wavelength on that path for each considered connection between a pair of nodes in such a way to maximize the total number of established connections in the network.

Marković and his coauthors in [33] had successfully solved this problem by the BCO meta-heuristic. They proposed the BCO heuristic algorithm tailored for the RWA problem. They called the proposed algorithm the BCO-RWA algorithm. The authors created the artificial network shown in the Fig.7.

The node depicted by the square in the Fig.7 represents hive. At the beginning of the search process all artificial agents are located in the hive. Bees depart from the hive and fly through the artificial network from the left to the right. Bee's trip is divided into stages. Bee chooses to visit one artificial node at every stage. Each stage represents the collection of all considered origin-destination pairs. Each artificial node is comprised of an origin and destination linked by a number of routes. Lightpath is a route chosen by bee agent. Bee agent's entire flight is collection of established lightpaths. The authors determined in advance the number of bees *B* and the number of iterations *I* as a stopping criteria.

During forward pass every bee visits *n* stages (bee tries to establish *n* new lightpaths). That means *NC* was set to *n* where *n* was selected in such a way that $n<<m$, *m* representing the total number of requested lightpaths. At every stage a bee chooses among remaining artificial nodes (not previously selected ones). Sequence of the *n*

visited artificial nodes generated by the bee represents one partial solution of the problem considered. Bee is not always successful in establishing lightpath when visiting artificial node. Bee's success depends on the wavelengths' availability on the specific links. In this way, generated partial solutions differ among themselves according to the total number of established lightpaths.

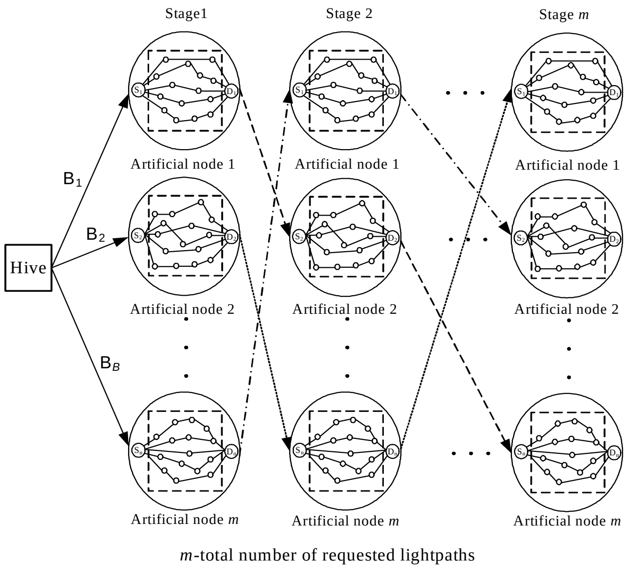

Fig. 7. Artificial network

Probability *p* that specific unvisited artificial node will be chosen by the bee equals $1/n_{unvis}$, where n_{unvis} is the total number of unvisited artificial nodes. By visiting specific artificial node in the network shown in Fig.7 bees attempt to establish the requested lightpath between one real source-destination node pair in optical network. Let us assume that the specific bee decided to consider the lightpath request between the source node *s* and the destination node *d*. In the next step, it is necessary to choose the route and to assign an available wavelength along the route between these two real nodes. In [33] for every node pair (*s, d*), the authors defined a subset R^{sd} of allowed routes that could be used when establishing the lightpath. These subsets were defined by using the *k* shortest path algorithm: For every of the *k* alternative routes the bee's utility when choosing the considered route is calculated. The shorter the chosen route and the higher the number of available wavelengths along the route, the higher the bee's utilities are. The authors define the bee's utilities V_r^{sd} when choosing the route *r* between the node pair (*s, d*) in the following way:

$$V_r^{s,d} = a\frac{1}{h_r - h_{r\min} + 1} + (1 - a)\frac{W_r}{W_{\max}} \tag{5}$$

where:

r – the route ordinary number for a node pair, r =1, 2,..., k, $r \in \{R^{sd}\}$;

h_r – the route length expressed in the number of physical hops;

$h_{r\min}$ – the length of the shortest route r_{min};

W_r – the number of available wavelengths along the route r;

$W_{\max} = \max_{r \in R^{sd}} \{W_r\}$ – the maximum number of available wavelengths among all routes $r \in R^{sd}$;

a – weight (importance of the criteria), $0 \le a \le 1$.

Bees decide to choose a physical route in optical network in a random manner. Inspired by the Logit model, the authors in [33] assumed that the probability p_r^{sd} of choosing route r in the case of origin-destination pair (s, d) equals:

$$p_r^{sd} = \begin{cases} \dfrac{e^{V_r^{sd}}}{\sum_{i=1}^{|R^{sd}|} e^{V_i^{sd}}} & \forall r \in R^{sd} \quad and \quad W_r > 0 \\ 0 & \forall r \in R^{sd} \quad and \quad W_r = 0 \end{cases} \tag{6}$$

where $|R^{sd}|$ is the total number of available routes between pair of nodes (s, d). The route r is available if there is at least one available wavelength on all links that belong to the route r.

After forward pass, bees perform backward pass, i.e. they return to the hive. In the hive every bee makes the decision about abandoning the created partial solution or expanding it in the next forward pass. The authors assumed that every bee can obtain the information about partial solution quality created by every other bee. The probability that the bee b would use the same partial tour that is defined in forward pass u, at the beginning of the u + 1 forward pass is calculated in the following way:

$$p_b = e^{-\frac{C_{\max} - C_b}{u}} \tag{7}$$

where:

C_b - the total number of established lightpaths from the beginning of the search process by the b-th bee;

C_{max} -the maximal number of established lightpaths from the beginning of the search process by any bee;

u - ordinary number of forward pass, u = 1,2,...

Let us discuss Eq. (7) that the authors propose in more details. Better generated partial solution (higher C_b value), implies the higher probability that the bee will be loyal to the previously discovered partial solution. Greater the ordinary number of the forward pass implies higher influence of the already discovered partial solution. This is expressed by the term u in the nominator of the exponent (Eq. (7)). In other words, at the beginning of the search process bees are "more brave" to search the solution space. The more forward passes they make, the bees have less courage to explore the solution space. The more we are approaching the end of the search process, the more focused the bees are on the already known solutions.

In [33] the probability p_P that the P-th advertised partial solution will be chosen by any of the uncommitted follower was calculated using the following relation:

$$p_P = \frac{e^{C_P}}{\sum_{p=1}^{P} e^{C_p}} \tag{8}$$

where C_P is the total number of the established lightpaths in the case of the P-th advertised partial solution.

The BCO-RWA algorithm was tested on a few numerical examples. The authors formulated corresponding Integer Linear Program (ILP) to determine optimal solutions for the considered examples. They compared the BCO-RWA results with the optimal solution. The comparison for the considered network is shown in the Table 2.

Table 2. The results obtained by comparison of BCO-RWA with ILP

Total number of requested light-paths	Number of wave-lengths	Number of established lightpaths		CPU time [s]		Relative error [%]
		ILP	BCO-RWA	ILP	BCO-RWA	
28	1	14	14	4	4.33	0
	2	23	23	94	4.58	0
	3	27	27	251	4.68	0
	4	28	28	313	4.66	0
31	1	15	14	4	4.73	6.67
	2	25	25	83	5.00	0
	3	30	30	235	5.19	0
	4	31	31	1410	5.21	0
34	1	15	14	14	5.19	6.67
	2	27	26	148	5.50	3.70
	3	33	33	216	5.64	0
	4	34	34	906	5.64	0
36	1	16	15	23	5.64	6.25
	2	27	26	325	6.09	3.70
	3	34	34	788	6.11	0
	4	36	36	1484	6.13	0
38	1	17	16	16	5.67	5.88
	2	28	27	247	6.09	3.57
	3	35	35	261	6.23	0
	4	38	38	1773	6.33	0
40	1	17	16	31	6.00	5.88
	2	28	27	491	6.28	3.57
	3	35	35	429	6.61	0
	4	40	40	1346	6.67	0

From the results presented in Table 2 it can be concluded that the proposed BCO-RWA algorithm has been able to produce optimal, or a near-optimal solutions in a reasonable amount of computer time.

4.4 BCO approach to optimize locations of traffic sensors on highways

The problem of the placement of point detectors within a roadway network belongs to the field of location theory. Point detectors are deployed on roadways to collect traffic data including volume, occupancy, and speed. The data is used by Traffic Management Centers in cities to manage traffic and incidents and provide information to motorists about current conditions. The spacing of detectors on freeways has a key impact on the travel time estimates obtained from the reported speeds. There is a tradeoff between detector spacing and travel time estimate correctness. As detectors become more closely spaced, the data obtained from them more closely look like continuous data available from probes. This additional accuracy also comes with much higher capital and ongoing costs, as all detectors require regular maintenance to continue to report good data. Transportation agencies are therefore seeking a method to indicate the most appropriate locations for detector deployment such that the travel time estimate error is minimized, within the constraints of available capital and maintenance funding.

Edara et al. in [20, 44] studied the problem of optimal placing traffic detectors on freeways and developed the BCO algorithm to solve it. The proposed model tries to minimize the error in travel time estimation, while taking into account the constraints of available capital and maintenance funding.

During the forward pass of the BCO algorithm the Logit model [34] was used for selection of the potential detector locations (*NC* was equal to one). The probability of a bee choosing a node *i* was expressed using the Logit model as follows:

$$p_i = \frac{e^{U_i}}{\sum\limits_{r=1}^{n} e^{U_r}} \qquad (9)$$

where U_i represented the utility of having a detector at node *i*. This utility depended on several factors that may affect travel time estimates. Factors such as the presence of a natural bottleneck at that location (e.g. a lane reduction) that leads to recurring congestion during the peak traffic periods, historical accident likelihoods (to monitor the induced delays by deploying detectors), level of traffic volumes, etc, can be used to determine the utilities. In [20, 44], it was assumed that all potential detector locations have equal utilities. Within each forward pass a bee visited a certain number of nodes and created a partial solution (choose few nodes that become detector locations).

Each generated partial solution in [20, 44] was characterized by the travel time estimation error. As the criteria for comparison of partial solutions, the maximum travel time error over all travel time runs was selected. By E_b the authors denoted the maximum travel time error over all travel time runs in the case of the partial solution created by the *b*-th bee. It was normalized by the following formula:

$$O_b = \frac{E_{\max} - E_b}{E_{\max} - E_{\min}}, \qquad O_b \in [0,1] \quad b = 1,2,...,B \tag{10}$$

having that:

O_b - was normalized value of the maximum travel time error over all travel time runs for the partial solution created by the *b*-th bee

$E_{\max}$, $E_{\min}$ - represented maximum and minimum travel time error value over all partial solutions generated so far.

The probability that *b*-th bee (at the beginning of the new forward pass) is loyal to its previously discovered partial solution was expressed as follows:

$$p_b^{u+1} = e^{-\frac{O_{\max} - O_b}{u}} \quad b = 1,2,...B \tag{11}$$

where *u* represented the ordinary number of the forward pass (e.g., *u*=1 for first forward pass, *u*=2 for second forward pass).

A bee that does not want to expand its previously generated partial solution would go to the dancing area of the hive to find another bee(s) to follow. The probability that *b*'s partial solution would be chosen by any uncommitted bee in [20, 44] was equal to:

$$p_b = \frac{O_b}{\sum_{k=1}^{R} O_k} \tag{12}$$

where:

O_k - objective function value of the *k*-th advertised solution;

R - the number of recruiters.

The proposed BCO algorithm was tested on a real-world freeway segment in Virginia. One of the main purposes of developing proposed methodology was to generate tradeoff plots between the travel time error and the number of detectors which would give the optimal placement of detectors for different levels of available funding.

Tradeoff plots were generated by varying the actual number of detectors (*d*) from 2 to 20 in increments of 1. Results of the BCO runs from [44] are shown in Fig.8. For a given number of detectors, the obtained optimal placement would result in a travel time estimation error for each travel time run. The maximum error versus the detector deployment obtained by the Genetic algorithms (GA) is also plotted in Fig.8.

These results enablled savings of 30% as compared to the current deployment at 20 locations. The obtained results were very competitive when compared with the results of Genetic Algorithms achieved in previous study.

The developed method is intended for use at a planning level, to assist in determining where to deploy detectors in an area that currently has few or no detectors, or in determining which detectors need to be (or those that need not be) regularly maintained to obtain good travel time estimates in areas with dense detector deployment.

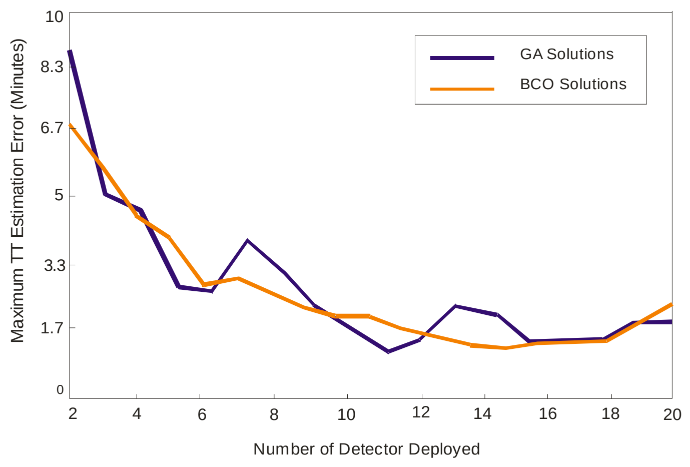

Fig. 8. Maximum Travel Time Estimation Error Plot (BCO vs GA)

4.5 Scheduling Independent Tasks by BCO and Parallel BCO

Davidović et al. [17, 18] applied BCO to the problem of static scheduling of independent tasks on identical machines. The problem can be described as follows. Let $T = \{1,2,...,n\}$ be a given set of independent tasks, and $P = \{1,2,...m\}$ set of identical machines. The processing time of task i ($i = 1,2,\ldots,n$) is denoted by l_i. All tasks are mutually independent and each task can be scheduled to any machine. All given tasks should be executed. Task should be scheduled to exactly one machine and machines can execute one task at a time. The goal is to find scheduling of tasks to machines in such a way as to minimize the completion time of all tasks (the so called *makespan*).

At each iteration of its execution BCO performs constructive steps composed of forward and backward passes and within them generates B solutions (schedules), one schedule for each bee. Within each forward pass every artificial bee is allowed to fly out from the hive and to generate NC task-machine pairs. The probability that specific bee chooses task I, denoted by p_i was calculated as follows:

$$p_i = \frac{l_i}{\sum_{k=1}^{K} l_k}, \; i=1,2,\ldots,n \tag{13}$$

where:

l_i – is the processing time of the i-th task;

K – represents the number of "free" tasks (not previously chosen).

Obviously, tasks with longer processing times have higher chances to be chosen. The probability p_j of choosing machine j by any bee equals:

$$p_j = \frac{V_j}{\sum_{k=1}^{m} V_k}, \; j=1,2,\ldots,m \tag{14}$$

where:

$$V_j = \frac{\max F - F_j}{\max F - \min F}, \; j=1,2,\ldots,m \tag{15}$$

F_j - running time of machine j based on tasks already scheduled to it;
max F, min F - maximum and minimum over all machines running times.

Machines with a lower value of the running times have a higher chance to be chosen. In total, B bees choose $B*NC$ task-machine pairs within each forward pass. After scheduling tasks to machines the corresponding machines' running times were updated.

After the completion of forward pass, all bees return to the hive and backward pass starts. Bees exchange information about the quality of the partial solutions generated. The latest time point of finishing the last task at any machine characterizes each generated partial solution. Upon obtaining full information about all partial solutions generated by all bees, every bee decides whether to abandon the food source and become again uncommitted follower, or dance and thus recruit the hive-mates before flying again from the hive and thus beginning the new forward pass. Forward and backward passes alternate until all bees generate the whole schedules.

If C_b (b=1, 2,..., B) denotes the latest time point of finishing the last task at any machine in the partial solution generated by the b-th bee, then O_b, the normalized value of the time point C_b, was calculated in [17, 18] in the following way:

$$O_b = \frac{C_{\max} - C_b}{C_{\max} - C_{\min}}, \quad b = 1,2,\ldots,B \tag{16}$$

where C_{min} and C_{max} are respectively the smallest and the largest time point among all time points produced by all bees. The probability that b-th bee (at the beginning of the new forward pass) is loyal to the previously discovered partial solution is calculated in this paper in the following way:

$$p_b^{u+1} = e^{-\frac{O_{\max} - O_b}{u}} \quad b = 1,2,\ldots B \tag{17}$$

where u is the ordinary number of the forward pass.

Ones the bee decided to stay loyal to its own partial solution, it is automatically becoming a recruiter, i.e. its solution is considered to be selected by any uncommitted bee (Fig. 9). The authors have assumed in [17, 18] that the probability the recruiter b's partial solution will be chosen by any uncommitted bee equals:

$$p_b = \frac{O_b}{\sum_{k=1}^{R} O_k} \tag{18}$$

where:

O_k - objective function value of the k-th advertised solution;

R - the number of recruiters.

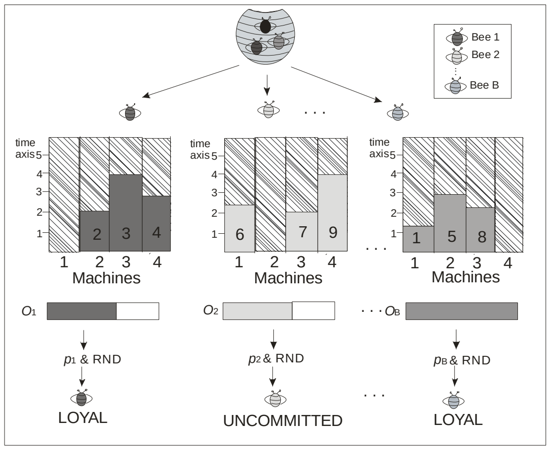

Fig. 9. Comparison of partial solutions after third forward pass, *NC*=1.

Using Eq. (18) and a random number generator, every uncommitted follower join one bee dancer (recruiter). Recruiters fly together with a recruted hive-mates in the next forward pass along the path discovered by the recruiter. At the end of this path all bees are free to independently search the solution space.

The proposed algorithm was tested on a various benchmark problems. Preliminary results were presented in [17], while the exhaustive experimental evaluations are described in [18]. The problem parameters range from instances with n = 100 up to the instances with n = 5000 and from p = 4 to p = 100. The BCO parameters were B = 5, NC = 10. The stopping criterion was the number of iterations and was equal to 100.

The authors compared the obtained BCO results with the optimal solution obtained by using ILOG AMPL and CPLEX 11.2 optimization software. The comparison results are illustrated in the Table 3. Within this table, the number of machines, m, is given in the first column, OPT denotes the optimal makespan, OPT Time is the value of CPU time required by CPLEX for solving the corresponding problem example to optimality. BCO represents objective function value obtained by the BCO algorithm; BCO error denotes deviation of BCO solution from the optimum one, BCO time shows the time required by BCO algorithm to obtain its final solution. The BCO algorithm was able to obtain the optimal value of objective function in most of the test problems. The CPU times required to find the best solutions by the BCO were negligible. All tests were performed on Intel Core 2 Duo CPU E6750 on 2.66GHz with RAM=8 Gb under Linux Slackware 12, Kernel: 2.6.21.5, gcc version 4.1.2.

Table 3. The comparison of the BCO results the optimal ones for *n*=5000

m	OPT	OPT Time (sec)	BCO	BCO error %	BCO time (sec)
4	6844	1.112	6844	0.000	0.070
8	3422	6.113	3422	0.000	0.209
16	1711	9.786	1711	0.000	0.217
25	1095	30.288	1095	0.000	0.226
50	548	28.561	548	0.000	0.251
100	274	1130.310	277	1.095	0.560

Parallel BCO. The above described implementation represented good starting point for testing parallelization strategies of BCO method. In [16] two synchronous parallelization strategies of BCO were proposed. The parallel BCO search was implemented on distributed memory IBM HPC Linux Cluster Server+16H2 Dual Core Intel Processors on 2.33GHz/1333MHz with 4MB RAM, Ethernet 3rd Party e1350 SMC 8848M Switch Bundle. The C programming language with MPI communication library was used.

The proposed parallelization strategies were tested on a various problem instances, the same one that have been used in [18]. It allowed authors to easily compare sequential and parallel BCO versions and measurement of the performance for various parallelization strategies. The representative subset of test examples has been chosen, namely the hard test instances from [15] with *a priori* known optimal solutions and the largest size examples from [52] that require a significant CPU time to be solved.

The target architecture for parallelized BCO in [16] was homogeneous completely connected network of processors. One of them is responsible for the communication with user and is named *master*. It is usually marked as processor 0. The other *q*-1 processors are called *working processors* or *slaves*. Their marks are processor 1 up to processor *q*-1. Parallel versions of BCO are executing on all *q* processors, i.e. computations are assigned to master too. Completely connected topology containing *q*=5 processors is shown on Fig. 10. In the experiments presented in [16] the number of processors was changing from 2 to 12.

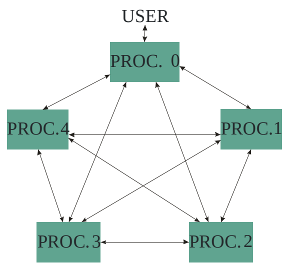

Fig. 10. Complete interconnection network of $q = 5$ processors

We present here some of the results for coarse grained parallelization strategy DBCO and for the other cases we just rewrite conclusions.

The results of scheduling one of the largest size examples from [52] (with 5000 tasks) on different number of machines are given in Table 4. These instances were not too hard to be solved by sequential BCO, and even they were solvable to optimality by CPLEX within a reasonable CPU time. For all examples, within DBCO parameter settings were the following: B=5, NC=10 and stopping criterion 1000 iterations.

Table 4. The comparison of the sequential and parallel BCO, results for n=5000

m	q	OPT	DBCO	DBCO time (sec)	S_q	E_q
4	1	6844	6844	60.04	1.00	1.00
	2		6844	31.83	1.89	0.94
	3		6844	21.36	2.81	0.94
	4		6844	15.99	3.75	0.94
	5		6844	12.76	4.71	0.94
8	1	3422	3422	61.94	1.00	1.00
	2		3422	32.70	1.89	0.94
	3		3422	21.86	2.83	0.94
	4		3422	16.34	3.79	0.95
	5		3422	13.07	4.74	0.95
16	1	1711	1711	65.65	1.00	1.00
	2		1711	34.77	1.89	0.94
	3		1711	23.20	2.83	0.94
	4		1711	17.38	3.78	0.94
	5		1711	13.89	4.73	0.95
25	1	1095	1095	69.75	1.00	1.00
	2		1095	37.12	1.88	0.94
	3		1095	24.76	2.82	0.94
	4		1095	18.50	3.77	0.94
	5		1095	14.12	4.70	0.94
50	1	548	548	81.03	1.00	1.00
	2		548	43.51	1.86	0.93
	3		548	29.04	2.79	0.93
	4		548	21.73	3.73	0.93
	5		548	17.41	4.65	0.93
100	1	274	277	104.46	1.00	1.00
	2		277	65.39	1.85	0.93
	3		277	37.54	2.78	0.93
	4		277	28.21	3.70	0.93
	5		277	22.56	4.63	0.93

The first column of the Table 4 contains the number m of machines within each example. The number of parallel processors q executing DBCO is given in the second column of our tables. Optimal schedule length represents the content of column three, while lengths of schedules obtained by DBCO for different q are placed in the next column. Column five in both tables contains CPU time required by DBCO to complete 1000 iterations, actually the CPU time required by q processor to complete 1000/q iterations. The corresponding speedup S_q and efficiency E_q are given in the last two columns. It is important to note that the CPU time required by DBCO to complete all necessary computations is actually the CPU time of the processor that is the last one to finish its work, i.e. it is equal to the maximum of all processors' running times. Actually, in the resulting tables we put the best obtained schedule length and the longest required CPU time.

Since for the calculation of the speedup and efficiency, “the best sequential algorithm” is required, in [16] it was assumed that BCO from [17, 18] can take the role of the best sequential algorithm. To assure fairness of obtained results, parallel versions of BCO were compared with the original sequential one executed on a single processor of given parallel architecture (instead of parallel version executed for q=1).

As can be seen from the results presented in Table 4 DBCO applied to those examples shows very good performance, almost linear speedup and above 90% efficiency, and also the stability in the solution quality (there is no degradation in parallel execution). In some other examples parallelization, solution quality was changing, the authors reported improvements or degradations of the solution quality for less than 3%.

When testing BBCO the authors obtained excellent (superlinar) speedup and efficiency, due to the reduction of computations assigned to each processor. On the other hand, FBCO resulted in slowing down the computations due to the communication delays caused by intensive data exchange between processors. This strategy is obviously more suitable for shared memory multiprocessor systems.

5 Conclusion

The Bee Colony Optimization, one of the newer Swarm Intelligence technique, is a meta-heuristic inspired by the foraging behavior of honeybees. It represents a general algorithmic framework applicable to various optimization problems in management, engineering, and control, and it should always be *tailored* for a specific problem. The BCO method is based on the concept of *cooperation*, which increases the efficiency of artificial bees and allows achievement of goals that could not be reached individually. The BCO has the capability, through the information exchange and recruiting process, to intensify the search in the promising regions of the solution space. When it is necessary, the BCO can also diversify the search. Recruited bees "fly together" with the recruiter along the path already generated by the recruiter. This means that partial solution generated by the recruiter is associated (copied) to recruited bees also. When they reach the end of the path, they are free to make an individual decision about the next constructive step to be made. The freedom to make an individual decision constitutes a diversifying element that complements the search intensification in the promising regions.

The BCO has already been successfully applied to several combinatorial optimization problems, and we hope that expanded application reports are to come soon. Moreover, the suitability for parallelization of the BCO algorithm opens not only a new research direction but also some new potential applications. However, the BCO has not been widely used for solving real-life problems and theoretical results supporting BCO concepts are still missing. This work is necessary in the future research. Based on the achieved results and gained experience, new models founded on BCO principles (autonomy, distributed functioning, self-organizing) are likely to significantly contribute to solving complex engineering, management, and control problems. Yet, the most important direction of the future research is the mathematical validation of the BCO approach. In years to come, the authors expect more BCO based models, examining, for instance, bees' homogeneity (homogenous vs. heterogeneous artificial bees), various information sharing mechanisms, and various collaboration mechanisms.

Acknowledgment

This research is partially supported by the Ministry of Science of Serbia, Grants No.144007 and 144033.

References

1. Abbass HA (2001) MBO: marriage in honey bees optimization-a Haplometrosis polygynous swarming approach. In: Proceedings of the Congress on Evolutionary Computation. Seoul, South Korea pp 207- 214
2. Afshar A, Bozorg Haddada O, Marin MA, Adams BJ (2007) Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. J. Frank. Instit. 344:452–462
3. Baykasoglu A, Özbakýr L, Tapkan P (2007) Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem. In: Felix TSC, Manoj KT (eds) Swarm Intelligence: Focus on Ant and Particle Swarm Optimization. Itech Education and Publishing, Vienna, Austria pp 113-143
4. Benatchba K, Admane L, Koudil M (2005) Using Bees to Solve a Data-Mining Problem Expressed as a Max-Sat One. In: Mira J, Alvarez JR (eds) IWINAC 2005, LNCS, 3562, Springer-Verlag Berlin Heidelberg pp 212–220
5. Beni G (1988) The concept of cellular robotic system. In: Proceedings of the 1988 IEEE International Symposium on Intelligent Control. IEEE Computer Society Press, Los Alamitos, CA, pp 57–62
6. Beni G, Hackwood S(1992) Stationary waves in cyclic swarms. In: Proceedings of the 1992 International Symposium on Intelligent Control. IEEE Computer Society Press, Los Alamitos, CA, pp 234–242
7. Beni G, Wang J (1989) Swarm intelligence. In: Proceedings of the Seventh Annual Meeting of the Robotics Society of Japan. RSJ Press, Tokyo, pp 425–428
8. Bonabeau E, Dorigo M, Theraulaz G (1997) Swarm Intelligence. Oxford University Press, Oxford

9. Brawer S (1989) Introduction to Parallel Programming. Academic Press, Inc
10. Camazine S, Sneyd J (1991) A Model of Collective Nectar Source by Honey Bees: Self-organization Through Simple Rules. J. Theor. Biol. 149:547-571
11. Chong CS, Low MYH, Sivakumar AI, Gay KL (2006) A Bee Colony Optimization Algorithm to Job Shop Scheduling Simulation. In: Perrone LF, Wieland FP, Liu J, Lawson BG, Nicol DM, Fujimoto RM (eds) Proceedings of the Winter Conference, Washington, DC pp 1954 – 1961
12. Crainic TG, Hail N (2005) Parallel meta-heuristics applications. In: Alba E (eds) Parallel Metaheuristics, John Wiley & Sons, Hoboken, NJ pp 447-494
13. Crainic TG, Toulouse M (2003) Parallel strategies for metaheuristics. In: Glover F, Kochenberger G (eds) Handbook in Metaheuristics, Kluwer Academic Publishers, pp 475-513
14. Cung VD, Martins SL, Ribeiro CC, Roucairol C (2002) Strategies for the parallel implementations of metaheuristics. In: Ribeiro CC, Hansen P (eds) Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, Norwell, MA pp 263-308
15. Davidović T, Crainic TG (2006) Benchmark problem instances for static task scheduling of task graphs with communication delays on homogeneous multiprocessor systems. Comput. Oper. Res. 33:2155-2177
16. Davidović T, Ramljak D, Šelmić M, Teodorović D (2010) Parallel Bee Colony Optimization for Scheduling Independet Tasks to Identical Machines, CCGrid2010 (submitted)
17. Davidović T, Šelmić M, Teodorović D (2009) Scheduling Independent Tasks: Bee Colony Optimization Approach. In: Proceedings of the 17th Mediterranean Conference on Control and Automation, MED'09, Thessaloniki, Greece pp 1020-1025
18. Davidović T, Šelmić M, Teodorović D (2009) Bee colony optimization for scheduling independent tasks to identical processors (submitted)
19. Drias H, Sadeg S, Yahi S (2005) Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem. In : Computational Intelligence and Bioinspired Systems, Lecture Notes in Computer Science 3512, Springer Berin/Heilderberg pp 318-325
20. Edara P, Šelmić M, Teodorović D (2008) Heuristic Solution Algorithms for a Traffic Sensor Optimization Problem, INFORMS 2008, Washington D.C.
21. Fathian M, Amiri B, Maroosi B (2008) A honeybee-mating approach for cluster analysis. Int. J. Adv. Manuf. Technol. 38: 809–821
22. Ferreira A, Morvan M (1997) Models for parallel algorithm design: An introduction. In: Migdalas A, Pardalos P, Storǿy S (eds) Parallel Computing in Optimization, Kluwer Academic Publishers, Dordrecht Boston London pp 1-26
23. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization (Technical Report-Tr06), Erciyes University, Engineering Faculty Computer Engineering Department Kayseri/Türkiye
24. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global. Optim. 39:459-471
25. Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl. Soft. Comput. 8:687–697
26. Karaboga D, Basturk Akay B, Ozturk C (2007) Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks. In: LNCS: Modeling Decisions for Artificial Intelligence, Springer-Verlag, Berlin Heidelberg pp 318-319
27. Kaufmann A, Gupta M (1988) Introduction to Fuzzy Arithmetic. New York: Van Nostrand Reinhold Company
28. Koudil M, Benatchba K, Tarabetand A, El Batoul Sahraoui (2007) Using artificial bees to solve partitioning and scheduling problems in codesign. Appl. Math. Comput. 186:1710-1722

29. Lučić P, Teodorović D (2001) Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence. In: Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis, Sao Miguel, Azores Islands, Portugal, pp 441–445
30. Lučić P, Teodorović D (2002) Transportation modeling: an artificial life approach. In: Proceedings of the 14th IEEE ''International Conference on Tools with Artificial Intelligence, Washington, DC, pp 216–223
31. Lučić P, Teodorović D (2003) Computing with bees: attacking complex transportation engineering problems. Int. J. Artif. Intell. T. 12: 375–394
32. Lučić P, Teodorović D (2003) Vehicle routing problem with uncertain demand at nodes: the bee system and fuzzy logic approach. In: Verdegay JL (eds) Fuzzy Sets in Optimization. Springer-Verlag, Heidelberg Berlin, pp 67–82
33. Marković G, Teodorović D, Aćimović-Rspopović V (2007) Routing and wavelength assignment in all-optical networks based on the bee colony optimization. AI Commun. 20:273–285
34. McFadden D (1973) Conditional Logit Analysis of Quantitative Choice Behavior. In: Zaremmbka P (eds), Frontier of Econometrics. Academic Press, New York
35. Navrat P (2006) Bee Hive Metaphor for Web Search. In: Rachev B, Smrikarov A, (eds) Proceedings of the International Conference on Computer Systems and Technologies – CompSysTech. Veliko Turnovo, Bulgaria, IIIA. pp 1-7
36. Pham DT, Ghanbarzadeh A, Koc E, Otri S, Zaidi M (2006) The Bees Algorithm - A Novel Tool for Complex Optimisation Problems. In: Proceedings of the 2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS 2006), Elsevier, Cardiff, pp 454-459
37. Pham DT, Soroka AJ, Ghanbarzadeh A, Koc E (2006) Optimising Neural Networks for Identification of Wood Defects Using the Bees Algorithm. In: Proceedings of the IEEE International Conference on Industrial Informatics, Singapore pp 1346-1351
38. Pham DT, Haj Darwish A, Eldukhr EE (2009) Optimisation of a fuzzy logic controller using the Bees Algorithm. Int. J., Comp. Aid. Eng. Tech. 1250 – 264
39. Porto SCS, Kitajima JPFW, Ribeiro CC (2000) Performance evaluation of a parallel tabu search task scheduling algorithm. Parallel Computing 26:73-90
40. Quijano N, Passino KM (2007) Honey Bee Social Foraging Algorithms for Resource Allocation, Part I: Algorithm and Theory. In: Proceedings of the 2007 American Control Conference, New York pp 3383-3388
41. Quijano N, Passino KM (2007) Honey Bee Social Foraging Algorithms for Resource Allocation, Part II: Application. In: Proceedings of the 2007 American Control Conference, New York pp 3389-3394
42. Quinn MJ (1987) Designing efficient algorithms for parallel computers. McGraw-Hill
43. Sato T, Hagiwara M (1997) Bee System: Finding Solution by a Concentrated Search. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics "Computational Cybernetics and Simulation". Orlando, FL, USA pp 3954-3959
44. Šelmić M, Edara P, Teodorović D (2008) Bee Colony Optimization Approach To Optimize Locations Of Traffic Sensors On Highways. Tehnika 6:9-15 (in Serbian)
45. Teodorović D (2003) Transport Modeling by Multi-Agent Systems: A Swarm Intelligence Approach. Transport. Plan. Techn. 26: 289–312
46. Teodorović D (2008) Swarm Intelligence Systems for Transportation Engineering: Principles and Applications. Transp. Res. Pt. C-Emerg. Technol. 16: 651-782
47. Teodorović D (2009) Bee Colony Optimization (BCO). In: Lim CP, Jain LC, Dehuri S (eds) Innovations in Swarm Intelligence. Springer-Verlag, Berlin Heidelberg pp 39-60
48. Teodorović D, Dell'Orco M (2005) Bee colony optimization – a cooperative learning approach to complex transportation problems. In: Advanced OR and AI Methods in Trans-

portation. Proceedings of the 10th Meeting of the EURO Working Group on Transportation, Poznan, Poland, pp 51–60

49. Teodorović D, Dell'Orco M (2008) Mitigating traffic congestion: solving the ride-matching problem by bee colony optimization. Transport. Plan. Techn. 31:135–152
50. Teodorović D, Lučić P, Marković G, Dell' Orco M (2006) Bee colony optimization: principles and applications. In: Reljin B, Stanković S (eds) Proceedings of the Eight Seminar on Neural Network Applications in Electrical Engineering – NEUREL 2006, University of Belgrade, Belgrade pp 151–156
51. Teodorović D, Šelmić M (2007) The BCO Algorithm For The *p* Median Problem. In: Proceedings of the XXXIV Serbian Operations Research Conference. Zlatibor, Serbia pp 417-420 (in Serbian)
52. Tobita T, Kasahara H (2002) A standard task graph set for fair evaluation of multiprocessor scheduling algorithms. J. Sched. 5:379-394
53. Todorović N, Petrović S, Teodorović D (2009) Bee Colony Optimization for Nurse Rostering (submitted)
54. Verhoeven MGA, Aarts EHL (1995) Parallel local search. J. Heur. 1:43-65
55. Wedde HF, Farooq M, Zhang Y. (2004) BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In: Ant Colony Optimization and Swarm Intelligence. LNCS 3172, Springer-Verlag, Berlin pp 83–94
56. Wedde HF, Timm C, Farooq M (2006) BeeHiveAIS: A Simple, Efficient, Scalable and Secure Routing Framework Inspired by Artificial Immune Systems. In: Runarsson TP et al. (eds) LNCS 4193, Springer-Verlag, Berlin Heidelberg pp 623–632
57. Wedde HF, Lehnhoff S, van Bonn B, Bay Z, Becker S, Böttcher S, Brunner C, Büscher A, Fürst T, Lazarescu M, Rotaru E, Senge, Steinbach B, Yilmaz F, Zimmermann T (2007) A Novel Class of Multi-Agent Algorithms for Highly Dynamic Transport Planning Inspired by Honey Bee Behavior. In: Proceedings of the 12th IEEE International Conference on Factory Automation, Patras, Greece pp 1157-1164
58. Yang C, Chen J, Tu X (2007) Algorithm of Fast Marriage in Honey Bees Optimization and Convergence Analysis. In: Proceedings of the IEEE International Conference on Automation and Logistics, Jinan, China pp 1794-1799
59. Yang X-S (2005) Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms. In: Mira J, Alvarez JR (eds) IWINAC 2005, Lecture Notes in Computer Science 3562, Springer-Verlag Berlin Heidelberg pp 317-323
60. Yonezawa Y, Kikuchi T (1996) Ecological algorithm for optimal ordering used by collective Honey bee behavior. In: Proceedings of the Seventh International Symposium on Micro Machine and Humane Science. Nagoya, Japan pp 249- 255
61. Zaheh L (1965) Fuzzy sets. Information and Control 8:338-353