Information Sciences 181 (2011) 4684–4698

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Honey bees mating optimization algorithm for the Euclidean traveling salesman problem

Yannis Marinakis [a,*], Magdalene Marinaki [b], Georgios Dounias [c]

[a] Technical University of Crete, Department of Production Engineering and Management, Decision Support Systems Laboratory, 73100 Chania, Greece
[b] Technical University of Crete, Department of Production Engineering and Management, Industrial Systems Control Laboratory, 73100 Chania, Greece
[c] University of the Aegean, Department of Financial and Management Engineering, Management and Decision Engineering Laboratory, 31 Fostini Str., 82100 Chios, Greece

ARTICLE INFO

Article history:
Available online 1 July 2010

Keywords:
Metaheuristics
Honey bees mating optimization
Traveling salesman problem

ABSTRACT

This paper introduces a new hybrid algorithmic nature inspired approach based on Honey Bees Mating Optimization for successfully solving the Euclidean Traveling Salesman Problem. The proposed algorithm for the solution of the Traveling Salesman Problem, the Honey Bees Mating Optimization (HBMOTSP), combines a Honey Bees Mating Optimization (HBMO) algorithm, the Multiple Phase Neighborhood Search-Greedy Randomized Adaptive Search Procedure (MPNS-GRASP) algorithm and the Expanding Neighborhood Search Strategy. Besides these two procedures, the proposed algorithm has, also, two additional main innovative features compared to other Honey Bees Mating Optimization algorithms concerning the crossover operator and the workers. The main contribution of this paper is that it shows that the HBMO can be used in hybrid synthesis with other metaheuristics for the solution of the TSP with remarkable results both to quality and computational efficiency. The proposed algorithm was tested on a set of 74 benchmark instances from the TSPLIB and in all but eleven instances the best known solution has been found. For the rest instances the quality of the produced solution deviates less than 0.1% from the optimum.

1. Introduction

Several biological and natural processes have been influencing the methodologies in science and technology in an increasing manner in the past years. Feedback control processes, artificial neurons, the DNA molecule description and similar genomics matters, studies of the behaviour of natural immunological systems, and more, represent some of the very successful domains of this kind in a variety of real world applications. During the last decade, nature inspired intelligence has become increasingly popular through the development and utilization of intelligent paradigms in advanced information systems design. Cross-disciplinary team-based thinking attempts to cross-fertilize engineering and life science understanding into advanced inter-operable systems. The methods contribute to technological advances driven by concepts from nature/biology including advances in structural genomics (intelligent drug design through imprecise data bases), mapping of genes to proteins and proteins to genes (one-to-many and many-to-one characteristics of naturally-occurring organisms), modelling of complete cell structures (showing modularity and hierarchy), functional genomics (handling of hybrid sources and heterogeneous and inconsistent origins of disparate databases), self-organization of natural systems, etc.[1] Among the

* Corresponding author. Tel.: +30 2821 037282; fax: +30 2821 069410.
E-mail addresses: marinakis@ergasya.tuc.gr (Y. Marinakis), magda@dssl.tuc.gr (M. Marinaki), g.dounias@aegean.gr (G. Dounias).
[1] EU's FP-6 Coordination Action NISIS: Nature Inspired Smart Information Systems (FP6–2002-IST-C) (http://www.nisis.de).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.06.032

most popular nature inspired approaches, when the task is optimization within complex domains of data or information, are those methods representing successful animal and micro-organism team behaviour, such as:

- Swarm or flocking intelligence: Birds flocks or fish schools have inspired Particle Swarm Optimization [42],
- Artificial immune systems: Algorithmic schemes that mimic the characteristics of the biological immune systems [14,15],
- Ant colonies: Ants' foraging behavior has given rise to Ant Colony Optimization [18], etc.

A number of nature inspired tools have been used to solve very diverse operations and supply chain management problems, like scheduling, organization of production and vehicle routing problems [68].

In the recent few years, various swarm intelligence algorithms, based on the behaviour of the bees have been presented [9]. These algorithms are mainly divided, in two categories according to their behaviour in the nature, the foraging behaviour and the mating behaviour. The most important approaches that simulate the foraging behaviour of the bees are the Artificial Bee Colony (ABC) Algorithm proposed by Karaboga and Basturk [40,41], the Virtual Bee Algorithm proposed by Yang [84], the Bee Colony Optimization Algorithm proposed by Teodorovic and DellOrco [77], the BeeHive algorithm proposed by Wedde et al. [83], the Bee Swarm Optimization Algorithm proposed by Drias et al. [19] and the Bees Algorithm proposed by Pham et al. [63].

The Artificial Bee Colony algorithm [40,41] is, mainly, applied in continuous optimization problems and simulates the waggled dance behaviour that a swarm of bees performs during the foraging process of the bees. In this algorithm, there are three groups of bees:

(1) The employed bees i.e. bees that determine the food source (possible solutions) from a prespecified set of food sources and share this information (waggle dance) with the other bees in the hive,
(2) The onlookers bees, i.e. bees that based on the information that they take from the employed bees, search for a better food source in the neighborhood of the memorized food sources, and
(3) The scout bees i.e. employed bees whose food source has been abandoned and a new food source search has started (randomly).

The Virtual Bee Algorithm [84] is, also, applied in continuous optimization problems. In this algorithm, the population of the bees is associated with a memory, a food source, and, then all the memories communicate between them with a waggle dance procedure. The whole procedure is similar with a genetic algorithm and it has been applied on two function optimization problems with two parameters. In the BeeHive [83] algorithm, a protocol inspired from dance language and foraging behaviour of honey bees is used. In the Bees Swarm Optimization [19], initially a bee finds an initial solution (food source) and from this solution the other solutions are produced with the use of certain strategies. Then, every bee is assigned in a solution and when the bees accomplish their search, they communicate between them with a waggle dance strategy and the best solution will become the new reference solution. To avoid cycling the authors use a tabu list. In the Bees Algorithm [63], a population of initial solutions (food sources) are randomly generated. Then, the bees are assigned to the solutions based on their fitness function. The bees return to the hive and based on their food sources a number of bees is assigned to the same food source in order to find a better neighborhood solution. In the Bee Colony Optimization [77] algorithm, a step by step solution is produced by each forager bee and when the foragers return to the hive a waggle dance is performed by each forager. Then the other bees, based on a probability, follow the foragers. This algorithm looks like the Ant Colony Optimization [18] algorithm but it does not use at all the concept of pheromone trails.

Contrary to the fact that there are many algorithms that are based on the foraging behaviour of the bees, the main algorithm proposed based on the marriage behaviour is the Honey Bees Mating Optimization algorithm (HBMO), that was presented in [1,2]. Since then, it has been used on a number of different applications [3,21,33,76]. The Honey Bees Mating Optimization algorithm simulates the mating process of the queen of the hive. The mating process of the queen begins when the queen flights away from the nest performing the mating flight during which the drones follow the queen and mate with her in the air [1,3]. The algorithm is a swarm intelligence algorithm since it uses a swarm of bees where there are three kinds of bees, the queen, the drones and the workers. There is a number of procedures that can be applied inside the swarm. In the Honey Bees Mating Optimization algorithm, the procedure of mating of the queen with the drones is described. First, the queen is flying randomly in the air and, based on her speed and her energy, if she meets a drone then there is a possibility to mate with him. Even if the queen mates with the drone, she does not create directly a brood but stores the genotype of the drone in her spermatheca and the brood is created only when the mating flight has been completed. With the term genotype we mean some of the basic characteristics of the drones, i.e. part of the solution. A crossover operator is used in order to create the broods. In a hive the role of the workers is simply the brood care (i.e. to feed them with the "royal jelly") and, thus, they are only a local search phase in the Honey Bees Mating Optimization algorithm. Thus, this algorithm combines both the mating process of the queen and one part of the foraging behavior of the honey bees inside the hive. If a brood is better (fittest) than the queen, then this brood replaces the queen.

In this paper, as there are not any competitive nature inspired methods based to Honey Bees Mating Optimization for the solution of the Traveling Salesman Problem (TSP), at least to our knowledge, we have decided to develop such an algorithm

and to test its efficiency compared to other nature inspired and classic metaheuristic algorithms. The proposed algorithm adopts the basic characteristics of the initially proposed Honey Bees Mating Optimization algorithm [1–3,21,33] and, also, makes a combined use of a number of different procedures in each of the subphases of the main algorithm in order to increase the efficiency of the proposed algorithm. More specifically, the proposed algorithm uses:

- The Multiple Phase Neighborhood Search-Greedy Randomized Adaptive Search Procedure (MPNS-GRASP) [56] for the calculation of the initial population of bees and of the initial queen. This procedure is used in order to have a more competitive queen.
- The Expanding Neighborhood Search (ENS) [54] as a local search strategy in order to have more effective and different workers. By using ENS, each brood has the possibility to select randomly the number of workers (local search phases) that will be used for the improvement of its solution.
- A new crossover operator based on an Adaptive Memory Procedure [70] and on a uniform crossover operator in order to have fittest broods. This crossover operator combines the genotype of the queen and of more than one drones to produce a brood. The reason why such a crossover operator is used is because in real life the queen stores in her spermatheca after the mating the genotype of all drones and after returning to the hive she produces the broods. The adaptive memory procedure is used in order to give the possibility to the queen to store from previous selected good drones (in previous mating flights) part of their solutions, for been able to use them in a new mating flight and for producing fittest broods.

The proposed algorithm was applied in the most classic combinatorial optimization problem, the Traveling Salesman Problem, with remarkable results since in most of the instances used, the algorithm found the best known solution. The proposed combination of procedures reduces, significantly, the computational time of the algorithm making the algorithm faster and more efficient and, thus, suitable for solving large-scale problems in short computational time. The rest of the paper is organized as follows: In the next section a description of the Traveling Salesman Problem is presented. In the third section the proposed algorithm, the Honey Bees Mating Optimization algorithm for the Traveling Salesman Problem (HBMOTSP) is presented and analyzed in detail. Computational results are presented and analyzed in the fourth section while in the last section conclusions and future research are given.

2. The traveling salesman problem

The Traveling Salesman Problem (TSP) is the problem of finding the shortest tour through all the cities that a salesman has to visit. The TSP is probably the most famous and extensively studied problem in the field of combinatorial optimization [32,45]. The TSP belongs to the class of NP-hard optimization problems [36]. Algorithms for solving the TSP may be divided into two classes, *exact algorithms* and *heuristic algorithms*. The exact algorithms are guaranteed to find the optimal solution in exponential number of steps. The most effective exact algorithms are branch and cut algorithms [44] with which large TSP instances have been solved [5]. The problem with these algorithms is that they are quite complex and are very demanding in computer power [35]. Heuristic attempts to solve the Traveling Salesman Problem are focused on tour construction methods [10,67,72] and tour improvement methods [24,48,49]. Tour construction methods build up a solution step by step while tour improvement methods start with an initial tour and, then, try to transform it into a shortest tour [37].

Since the introduction of Simulated Annealing [43] and Tabu Search [26,27] a breakthrough was obtained with the introduction of metaheuristics [29] which have the possibility to find their way out of local optima. For the solution of the TSP, a number of metaheuristic algorithms has been applied such as Simulated Annealing [11,62,67], Tabu Search [25,28,30,85], Genetic Algorithms [8,51,55,61,64–66,71,78,80,81], Variable Neighborhood Search [58], Iterated Local Search [5,37,60], Neural Networks [4,7,57,59,74], and Greedy Randomized Adaptive Search Procedure (GRASP) [54]. Finally, in the last fifteen years, a number of Nature Inspired or Swarm Intelligence Methods, like Ant Colony Optimization [6,12,16–18,20,46,50,53,75,79] and Particle Swarm Optimization [31,47,52,73,82] have been proposed for the solution of the TSP. A few years ago, an implementation challenge was organized by Johnson and McGeoch [38]. In this challenge, an effort was made to collect all possible algorithms that have been published in the area of the Traveling Salesman Problem. Also, this challenge gave the opportunity to researchers to present, test and compare their methods using the benchmark instances in the area. The results of the challenge were presented in two papers [38,39] and in the web page of the challenge (http://www.research.att.com/~dsj/chtsp/).

3. The honey bees mating optimization algorithm

3.1. General description of the algorithm

The proposed algorithm, the **Honey Bees Mating Optimization Algorithm for the Traveling Salesman Problem** (**HBMOTSP**), combines a number of different procedures. Each of them corresponds to a different phase of the mating

process of the honey bees. Initially, we have to choose the population of the honey bees that will configure the initial hive. There are two different ways to calculate the initial population, either completely at random (as in the initially proposed algorithm [1]) or by using an algorithm in order to obtain as good initial solutions as possible, and, thus, to obtain a more efficient queen. In the proposed algorithm, the initial population is created by using a modified version of the Greedy Randomized Adaptive Search Procedure (GRASP) [22,69], the Multiple Phase Neighborhood Search-GRASP (MPNS-GRASP) [54,56]. The MPNS-GRASP algorithm has been used in another hybrid algorithm proposed by Marinakis at al. [56] and the results obtained were significantly better than the results obtained when a random initial population was created. This is the reason why we used this method for the creation of the initial solution in the current problem. The best member of the initial population of bees is selected as the queen of the hive. All the other members of the population are the drones.

Before the process of mating begins, the user has to define a number that corresponds to the queen's size of spermatheca. This number corresponds to the maximum number of queen's mating in a single mating flight. Each time the queen successfully mates with a drone the genotype of the drone is stored and a variable is increased by one until the size of spermatheca is reached [2]. Another two parameters have to be defined, the number of queens and the number of broods that will be born by all queens. In this implementation of Honey Bees Mating Optimization (HBMO) algorithm, the number of queens is set equal to one as in the real life only one queen will survive in a hive, and the number of broods is set equal to the number corresponding to the size of queen's spermatheca. Then, the mating flight of the queen begins. At the start of the flight, the queen is initialized with some energy content (initially, the speed and the energy of the queen are generated at random) and returns to her nest when the energy is less than a threshold value (*thres*) and spermatheca is not full [3]. A drone mates with a queen probabilistically using the following annealing function [1,2]:

$$Prob(D) = e^{\left[\frac{-\Delta(f)}{Speed(t)}\right]} \tag{1}$$

where $Prob(D)$ is the probability of adding the sperm of drone D to the spermatheca of the queen (that is, the probability of a successful mating), $\Delta(f)$ is the absolute difference between the fitness of D and the fitness of the queen (for complete description of the calculation of the fitness function see below) and $Speed(t)$ is the speed of the queen at time t. The probability of mating is high when the queen is still at the beginning of her mating flight, therefore her speed is high, or when the fitness of the drone is as good as the queen's fitness. After each transition in space, the queen's speed and energy decay according to the following equations:

$$Speed(t+1) = \alpha \times Speed(t) \tag{2}$$

$$energy(t+1) = \alpha \times energy(t) \tag{3}$$

where α is a factor $\in(0,1)$ that determines the amount that the speed and the energy will be reduced after each transition and each step. It should be noted that Eq. (3) is different than the one proposed by Abbass [1,2] and it was introduced in this form as we would like to straightforwardly correlate the reduction of the speed with the reduction of the energy and, also, to use less parameters. Initially, the speed and the energy of the queen are generated at random. A number of mating flights are realized. At the start of a mating flight drones are generated randomly and the queen selects a drone using the probabilistic rule in Eq. (1). If the mating is successful (i.e. the drone passes the probabilistic decision rule), the drone's sperm is stored in the queen's spermatheca.

By using a crossover operator, a new brood (trial solution) is formed which later can be improved employing workers to conduct local search. A new crossover operator is developed in order to simulate the procedure that occurs in real life where the queen stores a number of different drones' sperm in her spermatheca and uses parts of the genotype of the different drones to create the new brood. Thus, the quality of the new solution (the brood) is fittest because as it takes parts of different solutions (queen and drones) it has more exploration abilities. This is a major difference of the proposed algorithm compared to other Honey Bees Mating Optimization algorithms [1,3,21,33,76] and to the classic evolutionary algorithms.

In real life, the role of the workers is restricted to brood care and for this reason the workers are not separate members of the population but they are used as local search procedures in order to improve the broods produced by the mating flight of the queen. Each of the workers has different capabilities and the choice of two different workers may produce different solutions. This is realized with the use of a number of single local search heuristics and their combinations. Thus, one of the parameters of the algorithm is the number of local search heuristics (w_1) and the other is the number of their combinations (w_2). The sum of these two numbers ($w = w_1 + w_2$) gives the number of workers. Each of the brood will choose, randomly, one worker to feed it with royal jelly (local search phase) having as a result the possibility of replacing the queen if the solution of the brood is better than the solution of the current queen. If the brood fails to replace the queen, then in the next mating flight of the queen this brood will be one of the drones if it has a better fitness function than one of the current drones. The number of drones remains constant in each iteration and the population of drones consists of the fittest drones of all

generations. A pseudocode of the proposed algorithm is presented in the following while in the next sections the procedures of the algorithm are explained in detail.

Algorithm 1: Honey Bees Mating Optimization for TSP

```
Initialization
Generate the initial population of the bees using MPNS-GRASP
Selection of the best bee as the queen
Selection of the maximum number of mating flights (M)
Main Phase
do while i ⩽ M
    Initialize queen's spermatheca, energy and speed.
    Select α
    do while energy > thres and spermatheca is not full
        Select a drone
        if the drone passes the probabilistic condition then
            Add sperm of the drone in the spermatheca
        endif
    Speed(t + 1) = α × Speed(t)
    energy(t + 1) = α × energy(t)
    enddo
    do j = 1, Size of Spermatheca
        Select a sperm from the spermatheca
        Generate a brood by applying a crossover operator between the queen,
            the selected drones and the adaptive memory
        Select, randomly, a worker
        Use the selected worker to improve the brood's fitness (ENS strategy)
        if the brood's fitness is better than the queen's fitness then
            Replace the queen with the brood
          else
            if the brood's fitness is better than one of the drone's fitness then
            Replace the drone with the brood
            endif
          endif
    enddo
enddo
return The Queen (Best Solution Found)
```

3.2. Initial population

GRASP [22,54,69] is an iterative two phase search method. Each iteration consists of two phases, a construction phase and a local search procedure. In the construction phase, a **randomized greedy function** is used to build up an initial solution. This randomized technique provides a feasible solution within each iteration. This phase can be described as a process which stepwise adds one element at a time to a partial (incomplete) solution. The choice of the next element to be added is determined by ordering all elements in a candidate list, the **Restricted Candidate List** (**RCL**), with respect to a greedy function. The probabilistic component of a **GRASP** is characterized by randomly choosing one of the best candidates in the list but not necessarily the top candidate. The greedy algorithm is a simple, one pass, procedure for solving the Traveling Salesman Problem. In the second phase, a **local search** is initialized from these points and the final result is simply the best solution found over all searches.

The most important differences of **MPNS-GRASP** from the classic GRASP concerns the construction of the RCL list and the application of alternative greedy functions in each iteration instead of only one simple greedy function as in the classical approach. Moreover, in MPNS-GRASP a combination of greedy functions is, also, possible. The algorithm starts with one greedy function and if the results are not improving an alternative greedy function is used instead (a threshold value b_2 determines this procedure) [56]. The utilization of a simple local search in the second phase of the classical algorithm limits the chances of obtaining better solutions, thus, MPNS-GRASP uses the Expanding Neighborhood Search, which is a very flexible local search strategy (see Section 3.5) [54]. The pseudocode of the MPNS-GRASP algorithm is given below.

Algorithm 2: MPNS-GRASP

Initialization
Select the set of the algorithms which will be used in the Main Phase 1 of the MPNS-GRASP
Select the set of local search algorithms which will be used in Main Phase 2 of the MPNS-GRASP
Select the threshold value b_2
! b_2 is the acceptance quality of the solution using a specified tour construction method in
! Main Phase 1 of the algorithm after a prespecified number of iterations
Select the initial size of the RCL
Main Phase
Set the number of iterations equal to zero
do while maximum number of iterations has not been reached
  Main Phase 1
    Increase the iteration counter
    Construct the RCL
    Select randomly from the RCL the candidate element for inclusion to the partial solution
      Call a greedy algorithm
      if the number of iterations is equal to the prespecified number **then**
          if the quality of the best solution for Main Phase 1 is larger than the threshold b_2 **then**
              Choose another construction heuristic
          endif
      endif
  Main Phase 2
        Call Expanding Neighborhood Search
enddo
return The best solution

3.3. Calculation of fitness function

In TSP, the fitness of each individual is related to the route length of each circle. Since the problems that we deal with are minimization problems, if a feasible solution has a large objective function value then it is characterized as an unpromising solution candidate and, therefore, its fitness must be set to a small value. Reversely, a large fitness value must correspond to a solution with a low objective function value. A way to accomplish this, is to find initially the individual in the population with the maximum objective function value and to subtract from this value the objective function value of each of the other individuals. By doing this, the larger fitness value corresponds to the tour with the shorter length. Since the probability of selecting an individual for mating is related to its fitness and since the individual with the worst objective function value has fitness equal to zero, it will never be selected for mating. Therefore, in order to avoid its total exclusion, the fitness of all individuals in this population is incremented by one, resulting, thus in a worse individual of fitness one.

3.4. Crossover operator

We propose a crossover operator which initially identifies the common characteristics of the parent individuals and, then, copies them to the broods. This crossover operator is a kind of Adaptive Memory Procedure. Initially, the adaptive memory has been proposed by Rochat and Taillard [70] as a part of a Tabu Search metaheuristic for the solution of the vehicle routing problem. This procedure stores characteristics (paths in the Traveling Salesman Problem) of good solutions. Each time a new good solution has been found the adaptive memory is updated. In our case, in the first generation the adaptive memory is empty. In order to add a solution or a part of a solution in the adaptive memory there are three possibilities:

(1) The candidate for the adaptive memory solution is a previous best solution (queen) that has fitness function value at most 10% worse than the value of the current best solution.
(2) The candidate for the adaptive memory solution is a member of the population (drone) that has fitness function value at most 10% worse than the value of the current best solution.
(3) A path is common for the queen and for a number of drones.

More analytically, in this crossover operator, the points are selected randomly from the adaptive memory, from the selected drones and from the queen. Thus, initially two crossover operator numbers are selected (Cr_1 and Cr_2) that control the fraction of the parameters that are selected for the adaptive memory, the selected drones and the queen. If there are common parts in the solutions (queen, drones and adaptive memory) then these common parts are inherited to the brood, else the Cr_1 and Cr_2 values are compared with the output of a random number generator, $rand_i(0,1)$. If the random number is less or equal to the Cr_1 the corresponding value is inherited from the queen, if the random number is between the Cr_1 and the

Cr_2 then the corresponding value is inherited, randomly, from the one of the elite solutions that are in the adaptive memory, otherwise it is selected, randomly, from the solutions of the drones that are stored in spermatheca. Thus, if the solution of the queen is denoted by $q_i(t)$ (t is the iteration number), the solution in the adaptive memory is denoted by $ad_i(t)$ and the solution of the drone by $d_i(t)$, then, the solution of the brood $b_i(t)$ is given by:

$$b_i(t) = \begin{cases} q_i(t), & \text{if } rand_i(0,1) \leqslant Cr_1 \\ ad_i(t), & \text{if } Cr_1 < rand_i(0,1) \leqslant Cr_2 \\ d_i(t), & \text{otherwise.} \end{cases} \tag{4}$$

In each iteration, the adaptive memory is updated based on the best solution.

3.5. Workers – expanding neighborhood search

As it has already been mentioned, the workers are not separate members of the population but they are used as local search procedures in order to improve the broods produced by the mating flight of the queen. The local search method that is used in this paper is the Expanding Neighborhood Search [54]. Expanding Neighborhood Search (ENS) is a metaheuristic algorithm [54] that can be used for the solution of a number of combinatorial optimization problems with remarkable results. The main features of this algorithm are (a) the use of the Circle Restricted Local Search Moves Strategy, (b) the ability of the algorithm to change between different local search strategies and (c) the use of an expanding strategy. These features are explained in detail in the following.

In the **Circle Restricted Local Search Moves–CRLSM** strategy, the computational time is decreased significantly compared to other heuristic and metaheuristic algorithms because all the edges that are not going to improve the solution are excluded from the search procedure. This happens by restricting the search into circles around the candidate for deletion edges. A Circle Restricted Local Search Moves Strategy for a **2-opt trial move** [48] is presented. There are three possibilities based on the costs of the candidates for deletion and inclusion edges:

- If both new edges increase in cost, a 2-opt trial move can not reduce the cost of the tour (e.g., in Fig. 1 (Possibility A), for both new edges the costs $C2$ and $C4$ are greater than the costs $B2$ and A of both old edges).
- If one of the two new edges has cost greater than the sum of the costs of the two old edges, a 2-opt trial move, again, can not reduce the cost of the tour (e.g., in Fig. 1 (Possibility B), the cost of the new edge $C3$ is greater than the sum of the costs $A + B3$ of the old edges).

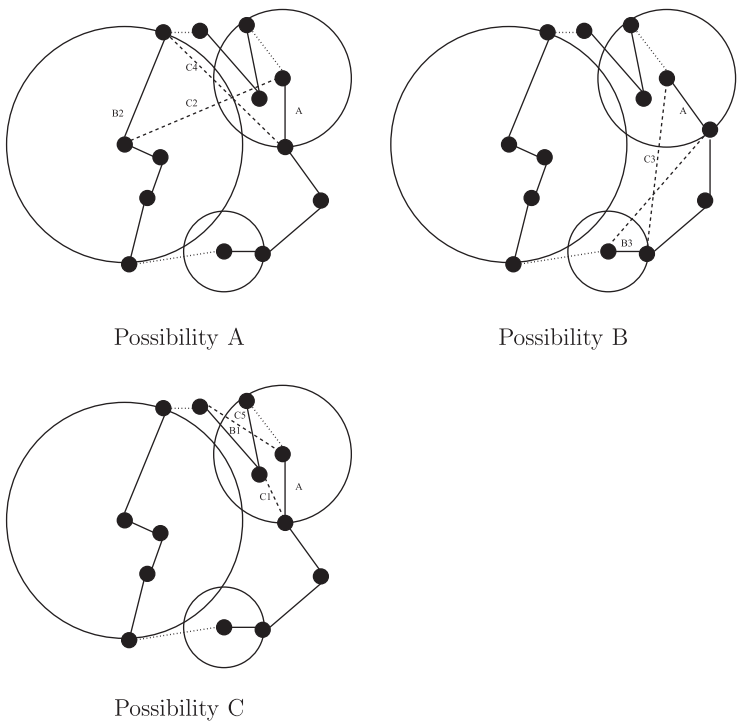

Fig. 1. Explanation of circle restricted local search moves strategy for a 2-opt trial move.

- The only case for which a 2-opt trial move can reduce the cost of the tour is when at least one new edge has cost less than the cost of one of the two old edges (e.g., in Fig. 1 (Possibility C), the cost $C1$ of the new edge is less than the cost of the old edge A) and the other edge has cost less than the sum of the costs of the two old edges (e.g., $C5 < A + B1$ in Fig. 1 (Possibility C)).

Taking these observations into account, the Circle Restricted Local Search Moves strategy restricts the search to edges where one of their end-nodes is inside a circle with radius length at most equal to the sum of the costs (lengths) of the two candidates for deletion edges.

The ENS algorithm has the ability to change between different local search strategies. The idea of using a larger neighborhood to escape from a local minimum to a better one had been proposed initially by Garfinkel and Nemhauser [23] and recently by Hansen and Mladenovic [34]. Garfinkel and Nemhauser proposed a very simple way to use a larger neighborhood. In general, if with the use of one neighborhood a local optimum was found, then a larger neighborhood is used in an attempt to escape from the local optimum. Hansen and Mladenovic proposed a more systematical method to change between different neighborhoods, called Variable Neighborhood Search.

On the other hand, the ENS method starts with one prespecified length of the radius of the circle of the CRLSM strategy. Inside this circle, different local search strategies are applied until all the possible trial moves have been explored and the solution can not further be improved in this neighborhood. Subsequently, the length of the radius of the circle is increased and, again, the same procedure is repeated until the stopping criterion is activated. The main differences of ENS from the other two methods is the use of the Circle Restricted Local Search Move Strategy which restricts the search in circles around the candidates for deletion edges and the more sophisticated way that the local search strategy can be changed inside the circles.

In ENS strategy, the size of the neighborhood is **expanded** in each external iteration [54]. Each different length of the neighborhood constitutes an external iteration. Initially, the size of the neighborhood, f, is defined based on the Circle Restricted Local Search Moves strategy, for example $f = A/2$, where A is the cost of one of the candidates for deletion edges. For the selected size of the neighborhood, a number of different local search strategies are applied until all the possible trial moves have been explored and the solution can not further be improved in this neighborhood. The local search strategies are changed based on two conditions, first if the current local search strategy finds a local optimum and second if the quality of the current solution remains greater than the threshold number b_1 for a number of internal iterations. (The quality is given in terms of the relative deviation from the best known solution). If the quality of the solution is less than the threshold number e the algorithm stops, otherwise the neighborhood is expanded by increasing the length of the radius of the CRLSM strategy f by a percentage θ (θ is determined empirically after thorough investigation and is set equal to 10%) and the algorithm continues. When the length of the radius of the CRLSM strategy is equal to A, the length continues to increase until the length becomes equal to $A + B$, where B is the length of the other candidate for deletion edge. If the length of the radius of the CRLSM strategy is equal to $A + B$, and the algorithm has reached the maximum number of iterations, then the algorithm terminates with the current solution. In Fig. 2, the Expanding Neighborhood Search Method is presented. The following pseudocode describes this approach.

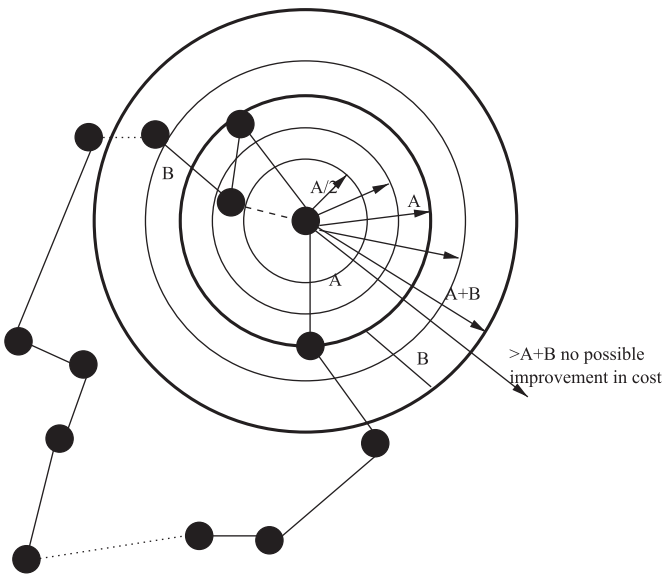

Fig. 2. Explanation of the expanding neighborhood search strategy for a 2-opt trial move.

Algorithm 3: Expanding Neighborhood Search

```
f = A/2, m = 0 !m = index for the local search methods
do while (ω > e or f < A + B)   !ω = the quality of the solution
    m = 1
    Call first local search strategy
    if ω < e then
      STOP with the current solution
    else
      do while (m < M1)
      !M1 = the number of local search strategy
          if ω < b1 or a local optimum is found then
              Change local search method
              m = m + 1
              if ω < e then
                  STOP with the current solution
              endif
          endif
      enddo
    endif
  f = 1.1* f
  Update b1 and e
enddo
```

In the Expanding Neighborhood Search strategy three local search algorithms are used. The **2-opt**, the **2.5-opt** and the **3-opt** algorithms which were introduced by Lin [48]. In the first algorithm, the neighborhood function is defined as exchanging two edges of the current solution with two other edges not in the current solution, in the second except of the exchanging of 2 edges there is additionally a node insertion move while in the third method the neighborhood function is defined as exchanging three edges of the current solution with three other edges not in the current solution.

4. Computational results

The whole algorithmic approach was implemented in Fortran 90 and was compiled using the Lahey f95 compiler on a Centrino Mobile Intel Pentium M 750 at 1.86 GHz, running Suse Linux 9.1. The parameters of the proposed algorithm were selected after thorough testing. A number of different alternative values were tested and the ones selected are those that gave the best computational results concerning both the quality of the solution and the computational time needed to achieve this solution. Thus, the selected parameters are: the number of queens is set equal to 1, the number of drones is set equal to 200, the number of mating flights (M) is set equal to 1000, the size of queen's spermatheca is set equal to 50, the number of broods is set equal to 50, the parameter α is set equal to 0.9, the number of different workers (w) is set equal to 7 ($w_1 = 3, w_2 = 4$), the size of RCL is set equal to 50 and, finally, the threshold value ($thres$) is set equal to 10^{-10}. It should be noted that the selection of the parameters used in the algorithm was performed taking into account a number of issues:

- The reason that the number of queens is set equal to one is that in real life in the hive only one queen exists.
- As in real life the queen mates with a number of drones and not with all of them, the spermatheca size should be smaller than the number of drones. Taking this into account we always keep the size of the spermatheca equal to 25% of the number of drones.
- In real life in each generation (mating flight) the number of broods could be different. However, as we would like to have an effective algorithm we chose the number of brood to be constant and equal to spermatheca's size in each mating flight.
- A number of values in the interval [10,5000] was tested for the number of drones. For these values the size of spermatheca was changed accordingly. For a large number of drones the algorithm needed a significant computational time without giving an improvement in the solutions, while when the number of drones was small the corresponding size of spermatheca was even smaller leading to a small diversity of the solutions.
- The number of mating flights was set equal to 1000 as for a larger number of mating flights the results were not improved.
- The parameter α was set equal to 0.9 as we would like to have a slow decrease of the queen's speed in order to perform more matings with the drones.

After the selection of the final parameters, 50 different runs with the selected parameters were performed for each of the benchmark instances.

Table 1
Results of the HBMOTSP.

Instance	HBMO-TSP	Optimum	ω_{av} (%)	ω (%)	CPU (s)	Instance	HBMO-TSP	Optimum	ω_{av} (%)	ω (%)	CPU (s)
Eil51	426	426	0	0	0.17	Rat575	6773	6773	0	0	57.18
Berlin52	7542	7542	0	0	0.19	P654	34643	34643	0	0	61.23
Eil76	538	538	0	0	0.53	D657	48912	48912	0	0	63.27
Pr76	108159	108159	0	0	0.53	Rat783	8806	8806	0	0	71.12
Rat99	1211	1211	0	0	0.58	dsj1000	18660556	18659688	0.012	0.0046	80.29
KroA100	21282	21282	0	0	0.62	Pr1002	259045	259045	0.001	0	80.57
KroB100	22141	22141	0	0	0.65	u1060	224094	224094	0	0	80.68
KroC100	20749	20749	0	0	0.63	vm1084	239297	239297	0.005	0	85.21
KroD100	21294	21294	0	0	0.64	Pcb1173	56892	56892	0.003	0	89.28
KroE100	22068	22068	0	0	0.61	D1291	50801	50801	0	0	91.14
Rd100	7910	7910	0	0	0.71	Rl1304	252948	252948	0	0	103.29
Eil101	629	629	0	0	0.87	Rl1323	270199	270199	0	0	113.43
Lin105	14379	14379	0	0	0.98	nrw1379	56638	56638	0.009	0	121.89
Pr107	44303	44303	0	0	1.01	Fl1400	20127	20127	0.011	0	198.67
Pr124	59030	59030	0	0	1.08	u1432	152270	152270	0.016	0	200.01
Bier127	118282	118282	0	0	1.11	Fl1577	22249	22249	0.022	0	227.28
Ch130	6110	6110	0	0	1.28	d1655	62149	62128	0.122	0.0338	241.67
Pr136	96772	96772	0	0	1.35	vm1748	336712	336556	0.189	0.0463	257.81
Pr144	58537	58537	0	0	1.68	u1817	57201	57201	0.028	0	289.12
Ch150	6528	6528	0	0	2.01	Rl1889	316536	316536	0.017	0	291.57
KroA150	26524	26524	0	0	2.12	D2103	80450	80450	0.041	0	350.78
Pr152	73682	73682	0	0	2.21	u2152	64267	64253	0.39	0.0217	357.23
Rat195	2323	2323	0	0	3.45	u2319	234256	234256	0.028	0	391.08
D198	15780	15780	0	0	4.27	Pr2392	378032	378032	0.026	0	401.28
KroA200	29368	29368	0	0	5.01	pcb3038	137694	137694	0.002	0	457.29
KroB200	29437	29437	0	0	4.99	fl3795	28783	28772	0.37	0.0382	461.81
Ts225	126643	126643	0	0	5.38	fnl4461	182612	182566	0.35	0.0251	498.01
Pr226	80369	80369	0	0	5.41	rl5915	565530	565530	0.012	0	557.87
Gil262	2378	2378	0	0	6.58	rl5934	556080	556045	0.0127	0.0062	561.21
Pr264	49135	49135	0	0	8.21	pla7397	23261400	23260728	0.0086	0.0028	780.31
A280	2579	2579	0	0	8.5	rl11849	923288	923288	0.098	0	890.05
Pr299	48191	48191	0	0	8.67	usa13509	19982859	19982859	0.087	0	897.09
Rd400	15281	15281	0	0	19.57	brd14051	469388	469388	0	0	902.35
Fl417	11861	11861	0	0	24.67	d15112	1573084	1573084	0.005	0	928.34
Pr439	107217	107217	0	0	36.21	d18512	645501	645244	0.25	0.039	995.17
Pcb442	50778	50778	0	0	37.12	pla33810	66079424	66050535	0.18	0.0437	1095.45
D493	35002	35002	0	0	45.21	pla85900	142495128	142383704	0.21	0.0782	1198.21

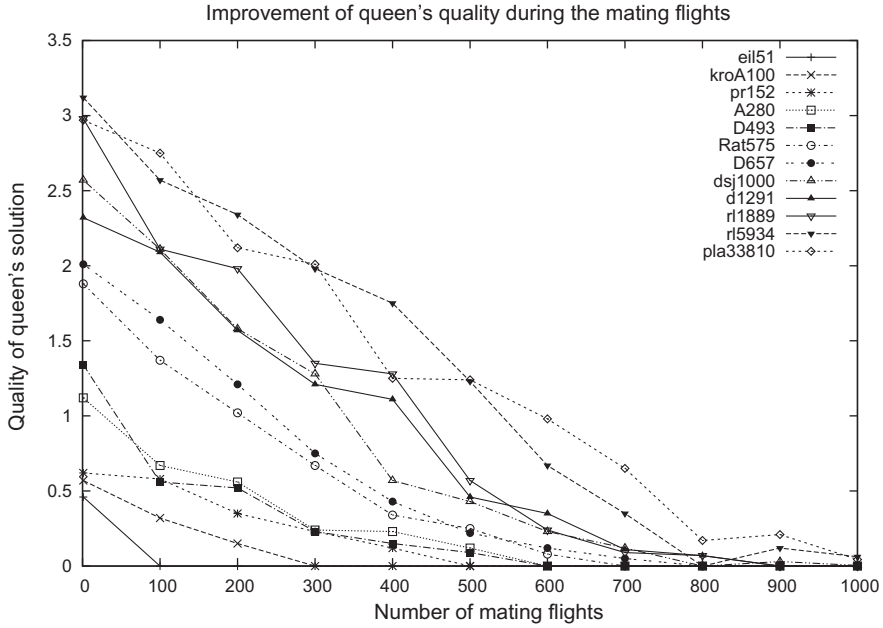

Fig. 3. Improvement of the queen's quality in a number of different instances during mating flights.

The algorithm was tested on a set of 74 Euclidean sample problems with sizes ranging from 51 to 85900 nodes. Each instance is described by its TSPLIB name and size, e.g., in Table 1 the instance named Pr76 has size equal to 76 nodes. In Table 1, the first column shows the name of the instance (the numbers in the names denote the nodes of each instance), the second column shows the solution from the HBMOTSP, the third column gives the optimal solution for each instance taken from the TSPLIB, the fourth column shows the average quality of the 50 runs of the algorithm (ω_{av}), the fifth column shows the quality of the best run of the proposed algorithm (ω) and finally, the sixth column gives the computational time (in seconds), of the best run, needed to find the solution. The quality [32] of the produced solutions is given in terms of the relative deviation from the optimum, that is $\omega = \frac{100(c_{\text{HBMOTSP}} - c_{opt})}{c_{opt}}$, where c_{HBMOTSP} denotes the cost of the optimum found by HBMOTSP, and c_{opt} is the cost of the optimal solution.

As it can be observed from this Table the optimal solution has been found in all but eleven instances, i.e. in 85.3% of all cases. For the other instances, the quality is less than 0.1%. Also, in this Table the computational time needed for finding the best solution by HBMOTSP is presented. The CPU time needed is low for the instances with number of nodes less than 1000. When the numbers of nodes is greater than 1000, the CPU time is somehow increased but still is very efficient. It should be noted that the aim of this study was to present a very fast and effective algorithm and, thus, the choice of the parameters (like the number of drones or the number of generations) was performed in such a way in order the algorithm to combine a fast convergence with as good as possible results. This issue in 11 out of 74 instances was the reason why the algorithm did not find the optimum. If we increase the number of drones and the number of generations the algorithm improves even more the solutions but then the algorithm finds these solutions in higher computational time. The difference in the quality of the best run's results from the average quality of the 50 runs is between 0.00% and 0.3683% while the average difference is equal to 0.029%. In all 50 runs of the algorithm, the best known solution was found in 46 instances. For the other instances, only in one or two runs the algorithm did not find the optimum. However, even if the best solution was not found in all runs, the solutions found were very close to the best solutions.

Table 2
Comparisons of the proposed algorithm with other algorithms (method 1 corresponds to ENS-GRASP, method 2 corresponds to MPNS-GRASP1, method 3 corresponds to MPNS-GRASP2, method 4 corresponds to HybGEN and method 5 corresponds to HBMOTSP).

Instance	Methods					Instance	Methods				
	1	2	3	4	5		1	2	3	4	5
Eil51	0	0	0	0	0	Rat575	1.32	0.23	0	0.68	0
Berlin52	0	0	0	0	0	P654	0.18	0.12	0	0.10	0
Eil76	0	0	0	0	0	D657	1.26	0.53	0	0.48	0
Pr76	0	0	0	0	0	Rat783	1.03	0.54	0	0.37	0
Rat99	0	0	0	0	0	dsj1000	1.17	0.28	0.0081	0.21	0.0046
KroA100	0	0	0	0	0	Pr1002	1.16	0.02	0	1.16	0
KroB100	0	0	0	0	0	u1060	1.01	0.44	0	0.27	0
KroC100	0	0	0	0	0	vm1084	1.18	0.34	0	0.51	0
KroD100	0	0	0	0	0	Pcb1173	1.37	0.04	0	1.57	0
KroE100	0	0	0	0	0	D1291	1.60	0.22	0	1.60	0
Rd100	0	0	0	0	0	Rl1304	0.88	0.40	0	0.88	0
Eil101	0	0	0	0	0	Rl1323	1.07	0.39	0	1.06	0
Lin105	0	0	0	0	0	nrw1379	1.21	0.33	0	0.82	0
Pr107	0	0	0	0	0	Fl1400	0.90	0.54	0	0.90	0
Pr124	0	0	0	0	0	u1432	1.07	0.66	0	0.71	0
Bier127	0.03	0	0	0	0	Fl1577	0.80	0.32	0	0.97	0
Ch130	0	0	0	0	0	d1655	1.21	0.67	0.0466	0.71	0.0338
Pr136	0	0	0	0	0	vm1748	1.25	0.62	0.0588	0.68	0.0463
Pr144	0	0	0	0	0	u1817	1.21	0.18	0	0.35	0
Ch150	0	0	0	0	0	Rl1889	0.85	0.23	0	0.85	0
KroA150	0	0	0	0	0	D2103	1.07	0.11	0	1.07	0
Pr152	0	0	0	0	0	u2152	1.12	0.48	0.0544	0.99	0.0217
Rat195	0.34	0	0	0	0	u2319	0.99	0.58	0	0.52	0
D198	0.05	0	0	0.04	0	Pr2392	2.11	0.29	0	1.62	0
KroA200	0.04	0	0	0.04	0	pcb3038	2.27	0.24	0	0.89	0
KroB200	0.15	0	0	0	0	fl3795	2.35	0.41	0.059	0.99	0.0382
Ts225	0	0	0	0	0	fnl4461	2.57	0.61	0.0597	1.02	0.0251
Pr226	0.05	0	0	0	0	rl5915	2.89	0.10	0	0.85	0
Gil262	0.29	0	0	0	0	rl5934	2.61	0.37	0.0071	0.87	0.0062
Pr264	0	0	0	0	0	pla7397	2.31	0.69	0.0033	0.75	0.0028
A280	0.38	0	0	0.04	0	rl11849	3.01	0.68	0	1.02	0
Pr299	0.09	0.09	0	0.09	0	usa13509	3.15	0.06	0	1.08	0
Rd400	0.68	0.68	0	0.35	0	brd14051	3.21	0.65	0	0.47	0
Fl417	0.28	0.28	0	0.24	0	d15112	2.95	0.57	0	0.34	0
Pr439	0.17	0.17	0	0.17	0	d18512	2.21	0.23	0.043	0.28	0.039
Pcb442	0.33	0.33	0	0.32	0	pla33810	3.01	0.20	0.0469	0.77	0.0437
D493	0.71	0.63	0	0.57	0	pla85900	3.29	0.73	0.0993	0.61	0.0782

In Fig. 3, the improvement of the queen's quality during different mating flights is presented. The value of the quality is given for every 100 mating flights. The results of twelve different instances, seven with number of cities less than 1000 and five with number of cities greater than 1000, are presented. As it can be observed for the instance eil51 the queen's quality became equal to optimum in only 100 mating flights. For the other instances with small number of cities the optimal solution was found in more mating flights but always in less than 700 mating flights. In the instances with a large number of nodes, the initial qualities were worse than the qualities of the other instances and, thus, more mating flights were needed to find the optimum.

In order to give the efficiency of the proposed algorithm and the role of each component of this algorithm, comparisons are performed with other metaheuristic algorithms (Table 2), namely with the Expanding Neighborhood Search-GRASP (ENS-GRASP) proposed by Marinakis et al. [54], with two versions of the Multiple Phase Neighborhood Search-GRASP (MPNS-GRASP1 and MPNS-GRASP2) proposed by Marinakis et al. [56] and with the Hybrid Genetic-GRASP (HybGEN) proposed by Marinakis et al. [55]. In ENS-GRASP the local search phase of the GRASP is the Expanding Neighborhood Search as it was described in Section 3.5. The MPNS-GRASP1 and MPNS-GRASP2 use more local search procedures than the ones used in this implementation of MPNS-GRASP. They use a variant of the well known Lin–Kernighan algorithm called Random Backtracking Lin–Kernighan (RBLK). The MPNS-GRASP2, also, uses a Path Relinking Process (PR). The reason why these two procedures have not been used in the HBMOTSP algorithm is that we would like to have a simplest version of MPNS-GRASP in order to emphasize more in the characteristics of the Honey Bees Mating Optimization algorithm. The MPNS-GRASP as applied in this paper is not complicated and it is very easy to be implemented. The HybGEN algorithm is a Hybrid Genetic algorithm where the ENS-GRASP is used in the mutation phase of a simple genetic algorithm in order to improve the solutions. The reason why HybGEN algorithm is used in the comparisons is because we would like to show the fact that HBMOTSP algorithm gives superior results compared to another population based algorithm. Another reason that we use the HybGEN is because in this algorithm it was proposed for the first time the crossover operator that we use in HBMOTSP. We can see from Table 2 that the results of the proposed algorithm are better from the results of the three (ENS-GRASP, MPNS-GRASP1 and HybGEN) out of the four algorithms. The results of the MPNS-GRASP2 algorithm are almost the same as in the proposed algorithm, i.e. both algorithms find in the same instances the optimum, but in the eleven instances where both algorithms did not find the optimum the results found by HBMOTSP are better in all instances from the MPNS-GRASP2. This is a very important conclusion of the comparisons because when was decided the use of MPNS-GRASP in the proposed algorithm without two basic components of the MPNS-GRASP2, the RBLK and the PR procedures, it was obvious that a more computationally efficient algorithm would be applied, but probably an algorithm that would not be able to give as good and competitive results as the ones of the MPNS-GRASP2. However, not only were found competitive results but in all instances were found better results in the HBMOTSP, from the MPNS-GRASP2. Thus, the use of the Honey Bees Mating Optimization

Table 3
Ranking of the algorithms based on the average quality (%).

Method	Average	Method	Average
Tour merging [5]	0.0034	3opt-J [37]	3.392
PHGA [61]	0.0066	TS-2opt-DB [85]	3.698
HBMO for TSP	0.0102	4-Hyperopt [37]	3.783
ILK-NYYY [38]	0.0354	GENIUS [24]	3.98
KH's MTV on LK [35]	0.0367	3-Hyperopt [37]	4.185
ILK-J [37]	0.0853	TS-2opt-2opt [85]	4.763
ALK [38]	0.48	2.5opt-B [10]	4.8585
Concorde CLK [5]	0.4863	2opt-J [37]	5.345
I-3-Opt-J [37]	0.514	2-Hyperopt [37]	5.386
ACR Chained LK [5]	0.5827	2opt-B [10]	6.091
ILK-N [60]	0.5924	HK-Christofides [38]	6.523
Helsgaun LK [35]	0.669	GENI [24]	7.88
BSDPH [38]	0.710	CCA [38]	10.291
LK-HK-Christo-starts [38]	1.051	CW [13]	10.594
TS-LK-DB [85]	1.060	FI [10]	14.623
TS-SC-DB [85]	1.111	2opt-C [5]	14.946
ENS-GRASP [54]	1.181	RI [10]	15.48
LK-NYYY [38]	1.228	C-Greedy [5]	16.741
Johnson LK [37]	1.388	Boruvka [5]	17.063
TS-LK-LK [85]	1.475	B-Greedy [38]	17.237
TS-SC-SC [85]	1.483	CHCI [10]	17.4185
Neto LK [60]	1.547	Q-Boruvka [5]	18.2487
SC EC [38]	1.660	NN [5]	24.916
VNS-3-Hyperopt [38]	2.061	Best - Way Strip [38]	48.0885
Concorde-LK [5]	2.778	Strip [38]	63.52
VNS-2-Hyperopt [38]	2.874	FRP [10]	64.3185
Applegate LK [5]	3.20	Spacefilling [38]	70.106
3opt-B [10]	3.288		

algorithm in a hybridization algorithm gives very good results. Concerning the HybGEN and the HBMOTSP it should be noted that although they both are population based algorithms and they both use the same crossover operator and ENS-GRASP, the HBMOTSP gives better results than the HybGEN.

The results of the proposed HBMOTSP algorithm are also compared with those presented in the DIMACS Implementation Challenge (http://www.research.att.com/~dsj/chtsp/). This challenge is probably the most extensive examination to date of heuristic algorithms in the field of TSP and presents computational results for approximately 40 tour construction heuristics and for approximately 80 tour improvement heuristics. In the challenge, the tour improvement heuristics are classified into four categories based on some specific characteristics. The first category consists of the classic local search algorithms, like 2-opt and 3-opt, while the second consists of algorithms that are different implementations of Lin–Kernighan algorithm. In the third category the algorithms that use Iterated or Chained Lin–Kernighan belong while in the last category the metaheuristics algorithms, like Tabu Search and Variable Neighborhood Search, are included. In Table 3, the ranking of all algorithms based on their average quality and for the instances with node number greater than 1000 is presented. The proposed HBMOTSP algorithm is ranked in the third place among 55 algorithms. The algorithm is ranked better than all tour construction heuristics. It is, also, ranked better than all simple local search algorithms. It is ranked better than all Variable Neighborhood Search implementations and it is, also, ranked better than all metaheuristics. For more details concerning the ranking of the algorithms please see [54]. It should be noted that a fair comparison in terms of computational efficiency is difficult because the computational speed is affected, mainly, by the compiler and the hardware that are used, thus the comparisons with the algorithms from the literature are performed only in terms of the quality of the solutions.

5. Conclusions and future research

In this paper, a nature inspired approach was introduced for the effective handling of the Euclidean Traveling Salesman Problem (TSP). More specifically, a hybrid algorithmic nature inspired methodology was proposed, namely the Honey Bees Mating Optimization algorithm for the TSP (HBMOTSP). One of the main contributions of this paper was to show that the Honey Bees Mating Optimization can be used in hybrid synthesis with other metaheuristics for the solution of the Traveling Salesman Problem with remarkable results both to quality and computational efficiency. The algorithm was applied on a set of benchmark instances from TSPLIB and gave very satisfactory results. Our future research will be mainly focused on the application of the Honey Bees Mating Optimization algorithm in other combinatorial optimization problems. We will also try to evolve the algorithm in order to obtain even better results by altering the role of workers and by using more than one queens and more than one hives.

References

[1] H.A. Abbass, A monogenous MBO approach to satisfiability, in: Proceeding of the International Conference on Computational Intelligence for Modelling, Control and Automation, CIMCA'2001, Las Vegas, NV, USA, 2001.
[2] H.A. Abbass, Marriage in honey-bee optimization (MBO): a haplometrosis polygynous swarming approach, in: The Congress on Evolutionary Computation (CEC2001), Seoul, Korea, May 2001, 2001, pp. 207–214.
[3] A. Afshar, O. Bozog Haddad, M.A. Marino, B.J. Adams, Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation, Journal of the Franklin Institute 344 (2007) 452–462.
[4] N. Ansari, E. Hou, Computational Intelligence for Optimization, first ed., Kluwer Academic Publishers, Boston, 1997.
[5] D. Applegate, R. Bixby, V. Chvatal, W. Cook, Chained Lin–Kernighan for large traveling salesman problems, Informs Journal on Computing 15 (2003) 82–92.
[6] A. Badr, A. Fahmy, A proof of convergence for ant algorithms, Information Sciences 160 (2004) 267–279.
[7] Y. Bai, W. Zhang, Z. Jin, An new self-organizing maps strategy for solving the traveling salesman problem, Chaos, Solitons and Fractals 28 (4) (2006) 1082–1089.
[8] R. Baralia, J.I. Hildago, R. Perego, A hybrid heuristic for the traveling salesman problem, IEEE Transactions on Evolutionary Computation 5 (6) (2001) 1–41.
[9] A. Baykasoglu, L. Ozbakor, P. Tapkan, Artificial bee colony algorithm and its application to generalized assignment problem, in: F.T.S. Chan, M.K. Tiwari (Eds.), Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, I-Tech Education and Publishing, 2007, pp. 113–144.
[10] J.L. Bentley, Fast algorithms for geometric traveling salesman problems, ORSA Journal on Computing 4 (1992) 387–411.
[11] Y. Chen, P. Zhang, Optimized annealing of traveling salesman problem from the *n*th-nearest-neighbor distribution, Physica A: Statistical and Theoretical Physics 371 (2) (2006) 627–632.
[12] S.C. Chu, J.F. Roddick, J.S. Pan, Ant colony system with communication strategies, Information Sciences 167 (1–4) (2004) 63–76.
[13] G. Clarke, J.W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research 12 (1964) 568–581.
[14] D. Dasgupta (Ed.), Artificial Immune Systems and Their Application, Springer, Heidelberg, 1998.
[15] L.D. De Castro, J. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach, Springer, Heidelberg, 2002.
[16] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.
[17] M. Dorigo, L.M. Gambardella, Ant colonies for the traveling salesman problem, Biosystems 43 (1997) 73–81.
[18] M. Dorigo, T. Stutzle, Ant Colony Optimization, A Bradford Book, The MIT Press Cambridge, Massachusetts, London, England, 2004.
[19] H. Drias, S. Sadeg, S. Yahi, Cooperative bees swarm for solving the maximum weighted satisfiability problem, in: IWAAN International Work Conference on Artificial and Natural Neural Networks, LNCS, vol. 3512, 2005, pp. 318–325.
[20] I. Ellabib, P. Calamai, O. Basir, Exchange strategies for multiple ant colony system, Information Sciences 177 (2007) 1248–1264.
[21] M. Fathian, B. Amiri, A. Maroosi, Application of honey bee mating optimization algorithm on clustering, Applied Mathematics and Computation 190 (2007) 1502–1513.
[22] T.A. Feo, M.G.C. Resende, Greedy randomized adaptive search procedure, Journal of Global Optimization 6 (1995) 109–133.
[23] R. Garfinkel, G. Nemhauser, Integer Programming, Wiley and Sons, 1972.
[24] M. Gendreau, A. Hertz, G. Laporte, New insertion and post optimization procedures for the traveling salesman problem, Operations Research 40 (1992) 1086–1094.

[25] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research 13 (1986) 533–549.
[26] F. Glover, Tabu search I, ORSA Journal on Computing 1 (3) (1989) 190–206.
[27] F. Glover, Tabu search II, ORSA Journal on Computing 2 (1) (1990) 4–32.
[28] Glover, F., 1990. Tabu search: A tutorial, Center for Applied Artificial Intelligence, University of Colorado, 1-47.
[29] F. Glover, G. Konchenberger, Handbook of Metaheuristics, Kluwer Academic Publishers, Dordrecht, 2003.
[30] F. Glover, M. Laguna, Tabu search, in: P.M. Pardalos, M.G.C. Resende (Eds.), Handbook of Applied Optimization, Oxford University Press, 2002, pp. 194–209.
[31] E.F.G. Goldbarg, G.R. Souza, M.C. Goldbarg, Particle swarm optimization for the traveling salesman problem, in: EVOCOP 2006, LNCS, vol. 3906, 2006, pp. 99–110.
[32] G. Gutin, A. Punnen, The Traveling Salesman Problem and its Variations, Kluwer Academic Publishers, Dordrecht, 2002.
[33] O.B. Haddad, A. Afshar, M.A. Marino, Honey-bees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization, Water Resources Management 20 (2006) 661–680.
[34] P. Hansen, N. Mladenovic, Variable neighborhood search: principles and applications, European Journal of Operational Research 130 (2001) 449–467.
[35] K. Helsgaum, An effective implementation of the Lin–Kernighan traveling salesman heuristic, European Journal of Operational Research 126 (2000) 106–130.
[36] D.S. Johnson, C.H. Papadimitriou, Computational complexity, in: E.L. Lawer, J.K. Lenstra, A.H.D. Rinnoy Kan, D.B. Shmoys (Eds.), The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley and Sons, 1985, pp. 37–85.
[37] D.S. Johnson, L.A. McGeoch, The traveling salesman problem: a case study, in: E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley and Sons, 1997, pp. 215–310.
[38] D.S. Johnson, L.A. McGeoch, Experimental analysis of heuristics for the STSP, in: G. Gutin, A. Punnen (Eds.), The Traveling Salesman Problem and its Variations, Kluwer Academic Publishers, Dordrecht, 2002, pp. 369–444.
[39] D.S. Johnson, G. Gutin, L.A. McGeoch, A. Yeo, W. Zhang, A. Zverovitch, Experimental analysis of heuristics for the ATSP, in: G. Gutin, A. Punnen (Eds.), The Traveling Salesman Problem and its Variations, Kluwer Academic Publishers, Dordrecht, 2002, pp. 445–487.
[40] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (2007) 459–471.
[41] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8 (2008) 687–697.
[42] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of 1995 IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
[43] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1982) 671–680.
[44] G. Laporte, The traveling salesman problem: an overview of exact and approximate algorithms, European Journal of Operational Research 59 (1992) 231–247.
[45] E.L. Lawer, J.K. Lenstra, A.H.G. Rinnoy Kan, D.B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley and Sons, 1985.
[46] F.X. Le Louarn, M. Gendreau, J.Y. Potvin, GENI ants for the traveling salesman problem, Annals of Operations Research 131 (2004) 187–201.
[47] X. Li, P. Tian, J. Hua, N. Zhong, A hybrid discrete particle swarm optimization for the traveling salesman problem, in: SEAL 2006, LNCS, vol. 4247, 2006, pp. 181–188.
[48] S. Lin, Computer solutions of the traveling salesman problem, Bell Systems Technical Journal 44 (1965) 2245–2269.
[49] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Operation Research 21 (1973) 498–516.
[50] A. Liu, Z. Deng, S. Shan, Mean contribution ant system: an improved version of ant colony optimization for traveling salesman problem, in: SEAL 2006, LNCS, vol. 4247, 2006, pp. 489–496.
[51] S.J. Louis, G. Li, Case injected genetic algorithms for traveling salesman problems, Information Sciences 122 (2000) 201–225.
[52] T. Machado, H. Lopez, A hybrid particle swarm optimization model for the traveling salesman problem, Adaptive and Natural Computing Algorithm (2005) 255–258.
[53] M. Manfrin, M. Birattari, T. Stutzle, M. Dorigo, Parallel ant colony optimization for the traveling salesman problem, in: ANTS 2006, LNCS, vol. 4150, 2006, pp. 224–234.
[54] Y. Marinakis, A. Migdalas, P.M. Pardalos, Expanding neighborhood GRASP for the traveling salesman problem, Computational Optimization and Applications 32 (2005) 231–257.
[55] Y. Marinakis, A. Migdalas, P.M. Pardalos, A hybrid Genetic-GRASP algorithm using Lagrangean relaxation for the traveling salesman problem, Journal of Combinatorial Optimization 10 (2005) 311–326.
[56] Y. Marinakis, A. Migdalas, P.M. Pardalos, Multiple phase neighborhood search GRASP based on Lagrangean relaxation and random backtracking Lin-Kernighan for the traveling salesman problem, Journal of Combinatorial Optimization 17 (2009) 134–156.
[57] T.A.S. Masutti, L.N. de Castro, A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, Information Sciences 179 (2009) 1454–1468.
[58] N. Mladenovic, P. Hansen, Variable neighborhood search, Computers and Operations Research 24 (1997) 1097–1100.
[59] A. Modares, S. Somhom, T. Enkawa, A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems, International Transactions in Operational Research 6 (6) (1999) 591–606.
[60] D.M. Neto, Efficient cluster compensation for Lin–Kernighan heuristics, Ph.D. Thesis, Computer Science University of Toronto, Canada, 1999.
[61] H.D. Nguyen, I. Yoshihara, K. Yamamori, M. Yasunaga, Implementation of an effective hybrid GA for large-scale traveling salesman problems, IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetic 37 (2007) 92–99.
[62] M. Ninio, J.J. Schneider, Weight annealing, Physica A: Statistical and Theoretical Physics 349 (3–4) (2005) 649–666.
[63] D.T. Pham, E. Kog, A. Ghanbarzadeh, S. Otri, S. Rahim, M. Zaidi, The bees algorithm – A novel tool for complex optimization problems, in: IPROMS 2006 Proceedings of Second International Virtual Conference on Intelligent Production Machines and Systems, Oxford, Elsevier, 2006.
[64] J.Y. Potvin, Genetic algorithms for the traveling salesman problem. Metaheuristics in combinatorial optimization, Annals of Operations Research 63 (1996) 339–370.
[65] L. Qu, R. Sun, A synergetic approach to genetic algorithms for solving traveling salesman problem, Information Sciences 117 (1999) 267–283.
[66] C. Rego, F. Glover, Local search and metaheuristics, in: G. Gutin, A. Punnen (Eds.), The Traveling Salesman Problem and its Variations, Kluwer Academic Publishers, Dordrecht, 2002, pp. 309–367.
[67] G. Reinelt, The Traveling Salesman, Computational Solutions for TSP Applications, Springer-Verlag, Berlin, 1994.
[68] J.F. Rennard, Handbook of research on nature inspired computing for economics and management, Idea Group Reference, Hershey-PA, USA, 2006.
[69] M.G.C. Resende, C.C. Ribeiro, Greedy randomized adaptive search procedures, in: F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, 2003, pp. 219–249.
[70] Y. Rochat, E.D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, Journal of Heuristics 1 (1995) 147–167.
[71] S. Ronald, Routing and scheduling problems, in: L. Chambers (Ed.), Practical handbook of genetic algorithms – Applications, CRC Press, New York, 1995, pp. 367–430.
[72] D.J. Rosenkratz, R.E. Stearns, P.M. Lewis, An analysis of several heuristics for the traveling salesman problem, SIAM Journal on Computing 6 (1977) 563–581.
[73] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu, Q.X. Wang, Particle swarm optimization-based algorithms for TSP and generalized TSP, Information Processing Letters 103 (2007) 169–176.

[74] P.H. Siqueira, M. Teresinha, A. Steiner, S. Scheer, A new approach to solve the traveling salesman problem, Neurocomputing 70 (4–6) (2007) 1013–1021.
[75] E.D. Taillard, Ant systems, in: P.M. Pardalos, M.G.C. Resende (Eds.), Handbook of Applied Optimization, Oxford University Press, 2002, pp. 130–138.
[76] J. Teo, H.A. Abbass, A true annealing approach to the marriage in honey bees optimization algorithm, International Journal of Computational Intelligence and Applications 3 (2) (2003) 199–211.
[77] D. Teodorovic, M. Dell'Orco, Bee colony optimization – A cooperative learning approach to complex transportation problems, Advanced OR and AI Methods in Transportation (2005) 51–60.
[78] C.K. Ting, S.T. Li, C. Lee, On the harmonious mating strategy through tabu search, Information Sciences 156 (2003) 189–214.
[79] C.F. Tsai, C.W. Tsai, C.C. Tseng, A new hybrid heuristic approach for solving large traveling salesman problem, Information Sciences 166 (2004) 6781.
[80] H.K. Tsai, J.M. Yang, Y.F. Tsai, C.Y. Kao, An evolutionary algorithm for large traveling salesman problems, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 34 (2004) 1718–1729.
[81] A. Ugur, Path planning on a cuboid using genetic algorithms, Information Sciences 178 (2008) 3275–3287.
[82] Y. Wang, X.Y. Feng, Y.X. Huang, D.B. Pu, W.G. Zhou, Y.C. Liang, C.G. Zhou, A novel quantum swarm evolutionary algorithm and its applications, Neurocomputing 70 (4–6) (2007) 633–640.
[83] H.F. Wedde, M. Farooq, Y. Zhang, BeeHive: an efficient fault-tolerant routing algorithm inspired by honey bee behavior, in: M. Dorigo (Ed.), Ant Colony Optimization and Swarm Intelligence, LNCS, vol. 3172, Springer, Berlin, 2004, pp. 83–94.
[84] X.S. Yang, Engineering optimizations via nature-inspired virtual bee algorithms, in: J.M. Yang, J.R. Alvarez (Eds.), IWINAC, LNCS, vol. 3562, Springer-Verlag, Berlin Heidelberg, 2005, pp. 317–323.
[85] M. Zachariasen, M. Dam, Tabu search on the geometric traveling salesman problem, in: I.H. Osman, J.P. Kelly (Eds.), Meta-Heuristics: Theory and Applications, Kluwer Academic Publishers, Boston, 1996, pp. 571–587.