Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2015) 23: 103 – 117

doi:10.3906/elk-1210-147

A hierarchic approach based on swarm intelligence to solve the traveling salesman problem

Mesut GÜNDÜZ[1], **Mustafa Servet KIRAN**[1,*], **Eren ÖZCEYLAN**[2]
[1]Department of Computer Engineering, Faculty of Engineering, Selçuk University, Konya, Turkey
[2]Department of Industrial Engineering, Faculty of Engineering, Selçuk University, Konya, Turkey

Received: 31.10.2012 • **Accepted:** 05.03.2013 • **Published Online:** 12.01.2015 • **Printed:** 09.02.2015

Abstract: The purpose of this paper is to present a new hierarchic method based on swarm intelligence algorithms for solving the well-known traveling salesman problem. The swarm intelligence algorithms implemented in this study are divided into 2 types: path construction-based and path improvement-based methods. The path construction-based method (ant colony optimization (ACO)) produces good solutions but takes more time to achieve a good solution, while the path improvement-based technique (artificial bee colony (ABC)) quickly produces results but does not achieve a good solution in a reasonable time. Therefore, a new hierarchic method, which consists of both ACO and ABC, is proposed to achieve a good solution in a reasonable time. ACO is used to provide a better initial solution for the ABC, which uses the path improvement technique in order to achieve an optimal or near optimal solution. Computational experiments are conducted on 10 instances of well-known data sets available in the literature. The results show that ACO-ABC produces better quality solutions than individual approaches of ACO and ABC with better central processing unit time.

Key words: Ant colony optimization, artificial bee colony, path construction, path improvement, hierarchic approach, traveling salesman problem

1. Introduction

The traveling salesman problem (TSP) is a classical benchmark NP-hard problem for discrete optimization techniques. The main objective of the TSP is to find the shortest Hamiltonian cycle that includes whole nodes [1]. Heuristic algorithms that try to find relatively good solutions in a reasonable time for this problem have been proposed, because there is no exact technique that finds the optimal solution in polynomial time [2]. Despite the fact that the TSP is very hard to solve, there are many applications of the TSP in real-world applications such as scheduling, assignment, and manufacturing problems [3].

Although a huge number of approaches by exact and heuristic techniques have been proposed by researchers for solving the TSP, some literature surveys by Langevin et al. [4], Laporte [5], Punnen [6], Bektaş [7], Rego et al. [8], Lawler et al. [9], Gutin and Punnen [10], and Applegate et al. [11] are presented for the readers. Branch and bound [12,13], cutting plane [14], branch and cut [15,16], and dynamic programming [17,18] techniques were developed for solving small instances of the TSP as exact methods. On the other hand, to yield acceptable solutions within a reasonable time, the genetic algorithm by Grefenstette et al. [19], Jog et al. [20], Qu and Sun [21], Larranaga et al. [22], Ray et al. [23], Liu et al. [24], Yang et al. [25], and Majumdar

*Correspondence: mskiran@selcuk.edu.tr

and Bhunia [26]; tabu search by Knox [27] and Gendreau et al. [28]; simulated annealing by Allwright and Carpenter [29], and Geng et al. [30]; neural networks by Ghaziri and Osman [31] and Leung et al. [32]; particle swarm optimization by Pang et al. [33,34], Wang et al. [35], Shi et al. [36], and Zhong et al. [1]; ant colony optimization (ACO) by Dorigo and Gambardella [37], Tsai et al. [38], Puris et al. [39], Bontoux and Feillet [40], and Puris et al. [41]; bee colony optimization by Wong et al. [42] and Marinakis et al. [43]; and artificial bee colony (ABC) optimization by Karaboğa and Görkemli [44] and Li et al. [45] have been applied. Aside from the above studies, several hybrid approaches have also been applied to TSPs. Particle swarm optimization and simulated annealing hybridization by Fang et al. [46], ACO and genetic algorithm hybridization by Takahashi [47], particle swarm optimization and ACO hybridizations by Gomez-Cabrero et al. [48] and Feng et al. [49], and genetic simulated annealing ant colony system with particle swarm optimization techniques by Chen and Chien [50] were developed so as to solve TSPs in a reasonable time. Table 1 summarizes the methods applied to TSPs.

Table 1. Methods applied to the TSP.

Methods	Authors
Branch and bound	Radharamanan and Choi [12], Singh and Oudheusden [13]
Cutting plane	Fleischmann [14]
Branch and cut	Padberg and Rinaldi [15], Perez and Gonzalez [16]
Dynamic programming	Chentsov and Korotayeva [17], Ergan and Orlin [18]
Genetic algorithm	Grefenstette et al. [19], Jog et al. [20], Qu and Sun [21], Larranaga et al. [22], Ray et al. [23], Liu et al. [24], Yang et al. [25], Majumdar and Bhunia [26]
Tabu search	Knox [27], Gendreau et al. [28]
Simulated annealing	Allwright and Carpenter [29], Geng et al. [30]
Artificial neural networks	Ghaziri and Osman [31], Leung et al. [32]
Particle swarm optimization	Pang et al. [33,34], Wang et al. [35], Shi et al. [36], Zhong et al. [1]
ACO	Dorigo and Gambardella [37], Tsai et al. [38], Puris et al. [39], Bontoux and Feillet [40], Puris et al. [41]
Bee colony optimization	Wong et al. [42], Marinakis et al. [43]
ABC	Karaboğa and Görkemli [44], Li et al. [45]
Particle swarm optimization and simulated annealing	Fang et al. [46]
ACO and genetic algorithm	Takahashi [47]
Particle swarm optimization and ACO	Gomez-Cabrero et al. [48], Feng et al. [49]
Genetic algorithm, simulating annealing, ant colony, and particle swarm optimization	Chen and Chien [50]

Most of these aforementioned heuristics fall into 1 of 2 categories: path construction heuristics and path improvement heuristics. A typical path construction method starts with a subset of points linked in a cycle and adds the others one by one until the cycle is complete. In this case, although the solution time is prolonged, the obtained solution is effectively acceptable. In contrast, path improvement heuristics work by taking a complete tour and repeatedly improving it. At each iteration, a number of possible changes are considered, and the best change found is made. The process continues until no change considered produces an improvement [2]. In this case, acceptable solutions could be quickly provided by the path improvement heuristic. In brief, path construction-based methods that produce good solutions take more time to achieve a good solution and path

improvement-based techniques that quickly produce results do not achieve a good solution in a reasonable time. For that reason, the initial solutions of path improvement heuristics can be generated through a more efficient construction heuristic. Herein lies the motivation of this study.

In this study, a new hierarchic method that consists of both ACO and ABC is proposed to achieve a better solution than the individual ABC and ACO algorithms in a reasonable time. In the hierarchic method, ACO is used as the path construction heuristic to obtain an acceptable feasible initial solution for the ABC, which is used as the path improvement heuristic. This study is separated from the existing studies by the following contributions: 1) describing a new hierarchic approach (ACO-ABC) that uses patch construction and improvement heuristics to solve TSPs, 2) comparing the obtained results with single patch construction (ACO) and improvement (ABC) heuristic applications, and 3) showing the superiority of ACO-ABC in all of the considered heuristic methods. The remainder of the paper is structured as follows. Following Section 1, which gives an introduction and literature survey on the TSP, Section 2 provides a formal description of the ACO and ABC algorithms and the proposed hierarchic approach is presented. Computational results and comparisons are presented in Section 3, and the results are discussed in Section 4. Finally, the conclusion and future works are given in Section 5.

2. Materials and methods

By combining the abilities of the path improvement and construction methods, a hierarchic optimization technique is proposed in order to obtain better quality results within a certain time. ACO is used as the path constructor and the results obtained by ACO are improved using the ABC. ACO and ABC are iterative methods; ABC only uses information in the population and ACO uses information in both the population (pheromone mechanism) and the problem (visibility) to achieve the global optimum for the optimization problems. In the hierarchic method, ACO is used for producing the initial solution and this solution is improved using ABC.

2.1. Ant colony optimization

Individual agents (called artificial ants) of ACO construct the self-solutions at each iteration. First, all of the artificial ants are randomly distributed to solution parts. In order to complete the self-solutions of the ants, each ant decides which part of the solution is selected in the next step using Eq. (1) [51].

$$p_{i,j}^{k}(t) = \begin{cases} \frac{\tau_{i,j}^{\alpha}(t) \times \eta_{i,j}^{\beta}}{\sum_{s \in N} \tau_{i,s}^{\alpha}(t) \times \eta_{i,s}^{\beta}} & \text{if}(s \in N) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Here, $p_{i,j}^{k}(t)$ is the selection probability of the jth solution part by the kth ant on the ith solution part, $\tau_{i,j}(t)$ is the quantity of the pheromone between the ith and jth solution parts at time t, $\eta_{i,j}$ is the visibility value between the ith and jth solution parts and is calculated using Eq. (2), α and β are significant factors used for tuning the weight of the pheromone and visibility, and N is a set of unused solution parts.

$$\eta_{i,j} = \frac{1}{\phi_{i,j}} \tag{2}$$

Here, $\phi_{i,j}$ is the cost of the (i,j) part. After all of the artificial ants complete the self-solutions using Eq. (1), the pheromone between the solution parts is evaporated and then laid using Eq. (3) [51].

$$\tau_{i,j}(t+1) = \tau_{i,j}(t) \times (1-p) + \sum_{k=1}^{n} \Delta\tau_{i,j}^{k}(t) \tag{3}$$

Here, p is the evaporation rate between (0,1], n is the number of artificial ants, and $\Delta\tau_{i,j}^{k}(t)$ is the pheromone quantity to be laid for the (i,j) solution part at time t and is calculated as follows [51]:

$$\Delta\tau_{i,j}^{k}(t) = \begin{cases} \frac{Q}{d_k} & \text{if}(k^{th}\,\text{ant used}\,(i,j)\,\text{solution part in self solution}) \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

where d_k is the cost of the kth ant solution and Q is a constant number. ACO is an iterative algorithm and the maximum cycle number, minimum error rate, etc. can be given as a termination condition to ACO. Based on the explanations given above, the ACO algorithmic framework is displayed in Table 2.

Table 2. Algorithmic framework of the ACO.

Step 1. *Algorithm initialization*
Determine number of artificial ants.
Create pheromone matrix.
Load solution space of the problem.
Step 2. *Solution initialization*
Distribute all the ants randomly.
Step 3. *Solution construction*
For all ants
While (Solution is not completed)
Select next solution part using Eq. (1).
End while
End for
Step 4. *Pheromone mechanism*
Evaporate and lay pheromone using Eq. (3).
Step 5. *Termination condition*
If a termination condition is not meet, return to Step 2.
Step 6. *Finalization*
Report the best results obtained.

2.2. Artificial bee colony algorithm

The ABC was first proposed to solve numerical optimization problems by Karaboğa [52]. Some discrete versions of the algorithm were developed and applied to different discrete optimization problems [53–55]. Kıran et al. [55] showed its performance and accuracy for TSPs and for identifying which neighborhood operator (swapping, insertion, etc.) is better than others.

We use a discrete version of the ABC algorithm to solve the TSP and the neighborhood operator by Kıran et al. [55] produces the best results in the hierarchic approach.

In the basic ABC, there are 3 kinds of bees: employed, onlooker, and scout bees. The number of employed bees is equal to the number of onlooker bees and only one scout bee can occur at each iteration. Employed bees have a self-solution and try to improve the self-solution and move information about food source positions to the hive. Onlooker bees do not have a self-solution, but they use the solutions of the employed bees for improving the solution by taking advantage of information in the hive. The scout bee occurrence is controlled by a peculiar parameter called the limit, and if a solution of the employed bee cannot improve within a certain time (limit), the employed bee of this solution becomes a scout bee. In the ABC, positions of food sources represent the feasible solution for the optimization problems.

First, all of the employed bees are distributed to the solution space of the optimization problem using Eq. (5) [52] and their abandonment counters are reset.

$$x_i^j = x_{\min}^j + rand[0,1] * (x_{\max}^j - x_{\min}^j), \text{ for } all j = 1, 2, \ldots, D \tag{5}$$

Here, x_i^j is a parameter to be optimized for the ith employed bee on dimension j of the D-dimensional solution space, and $x_{\max}^j$ and $x_{\min}^j$ are the upper and lower bounds for x_i^j, respectively.

Next, new food sources are produced for all of the employed bees using Eq. (6) [52].

$$v_{i,j} = x_{i,j} + \Phi \times (x_{i,j} - x_{k,j}) \quad j \in \{1, 2, ..., D\}, \quad k \in \{1, 2, ..., n\} \text{ and } i \neq k \tag{6}$$

Here x_i is the ith employed bee, v_i is the candidate solution for x_i, x_k is an employed neighbor bee of x_i, Φ is a number randomly selected in the range of [–1,1], n is the number of employed bees, D is the dimensionality of the problem, and $j \in \{1, 2, ..., D\}$ and $k \in \{1, 2, ..., n\}$ are randomly selected from the dimensionality of the problem and the employed bee population, respectively. In addition, only one parameter of the employed bee is updated at the each iteration.

After a new solution is produced, the new and old solutions are compared using the fitness values of the solutions, calculated as follows [52]:

$$fit_i = \begin{cases} \frac{1}{1+f_i} & \text{if}(f_i \geq 0) \\ 1 + abs(f_i) & \text{if}(f_i < 0) \end{cases}, \tag{7}$$

where f_i is the specific object function value for the problem. If the fitness value of the new solution is better than that of the old one, it is replaced and the abandonment counter of the new solution is reset; otherwise, the abandonment counter of the employed bee is increased by 1.

The onlooker bees produce a new solution by taking advantage of the position information about food sources shared by the employed bees in the hive. An onlooker bee selects an employed bee in order to improve its solution using Eq. (8) [52] and a roulette wheel selection. Next, the new food source position is produced using Eq. (6). If the fitness value of the onlookers' solution is better than the fitness value of the employed bee, it is replaced and the employed bees' abandonment counter is reset; otherwise, the counter is increased by 1.

$$p_i = \frac{fit_i}{\sum_{j=1}^{n} fit_j} \tag{8}$$

Here p_i is the probability to be selected for the ith employed bee or food source.

The scout bee of the algorithm is used for the global search and for getting rid of local minima. If the solution of the employed bee cannot improve until the abandonment counter of employed bee achieves the limit, the employed bee becomes a scout bee and a new solution is produced using Eq. (5), and the abandonment counter of the employed bee is reset. There is an important point in the occurrence of scout bees. At each iteration, only one employed bee that has the highest content of abandonment counters of the employed bees can be a scout bee. The algorithmic framework of the ABC is given in Table 3.

After some modifications, discrete optimization problems can be solved using the ABC. The first modifi-

Table 3. Algorithmic framework of the ABC.

Step 1. *Algorithm initialization*
Determine number of artificial bees (half of the population is employed, and the other half are onlookers).
Generate initial solutions for the employed bees using Eq. (5).
Reset the abandonment counters of the employed bees.
Determine the limit value.
Step 2. *Employed bee phase*
For all employed bees
Generate new food source using Eq. (6).
Calculate fitness value of the new solution using Eq. (7).
If new fitness value is better than the old, replace it; else increase abandonment counter by 1.
End for
Step 3. *Onlooker bee phase*
Calculate the probabilities to be selected of the employed bees using Eq. (8).
For all onlooker bees
Select an employed bee using Eq. (8) and roulette wheel.
Generate new food source using Eq. (6).
Calculate fitness value of new solution using Eq. (7).
If new fitness value is better than the old, replace it; else increase abandonment counter by 1.
End for
Step 4. *Scout bee phase*
Fix the abandonment counter H with the highest content.
IF the content of counter H is higher than the predefined limit THEN generate new solution for the employed bee to which counter H belongs using Eq. (5) and reset the abandonment counter of the employed bee; ELSE continue.
Step 5. *Termination condition*
If a termination condition is not meet return to Step 2.
Step 6. *Finalization*
Report the best results obtained.

cation is to change Eq. (6), which is used for obtaining new candidate solutions. In the discrete version of the ABC, Eq. (6) is replaced with neighborhood operators. Which neighborhood operator is better than the other was shown in [55]. According to the mentioned study, we use a combined operator (random insertion, random insertion of subsequence, and random reversing insertion of subsequence) in order to obtain a new candidate solution in the employed and onlooker bee phases of the ABC. The second modification is the initialization of the algorithm, and random permutations are given to the employed bees as initial solutions instead of using Eq. (5). The third important point in the discrete ABC is to select a high value for the limit due to the fact that scout bees prevent achieving saturation of the population for discrete optimization problems [55]. For the TSP, the object function is a tour length of the solution, and we try to find the global minimum for this problem.

2.3. ACO-ABC hierarchic approach

Stigmergy can be defined as the guidance of work in a progress or indirect communication among workers for cooperation and coordination among social insects [56]. In ACO, the pheromone mechanism helps to find tours with quality, but the distance between the nodes can cause stagnation for the agents because a larger pheromone trail will be between the close nodes. In the ABC, if a food source position cannot be improved by employed or onlooker bees within a certain time, this food source is abandoned. Therefore, the food source (work) guides the artificial agents (workers) in performing the labor. In order to overcome stagnation of the artificial ants in ACO, we propose a hierarchic approach based on ACO and ABC. Though ACO stagnates after a certain number of iterations, the ABC can improve the solution obtained by ACO. The ACO algorithm constructs the solution step by step and the number of artificial agents in ACO generally equals the number of nodes in the TSP. In practice, we show that ACO cannot improve the solution after a number of iterations, and this situation (stagnation behavior) is shown Figure 1. Therefore, we give the best solution obtained by ACO to the employed bees of the ABC as the initial solution after a certain number of iterations. We try to improve the best solution taken from ACO using a discrete ABC in order to increase the quality of the solution. By doing this, we obtain slightly better solutions than those of ACO in less time and much better solutions than those of the ABC.

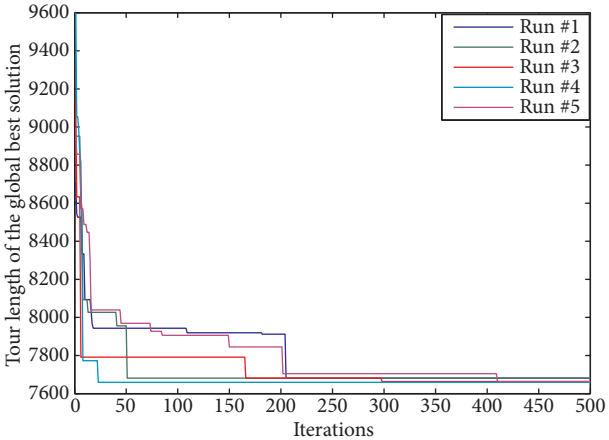

Figure 1. Stagnation behavior of the ACO on Berlin52 TSP.

As seen from Figure 1, none of the runs provide improvements until the maximum cycle number is reached. After the 250th cycle, the global best solution does not generally show any alteration in ACO, although a good solution is obtained for the problem. Hence, we stop the ACO at half of the maximum number of cycles and

transfer the global best solution obtained by ACO to the ABC as the initial solution. The scheme and the working diagram of the hierarchic approach are shown in Figures 2 and 3, respectively.

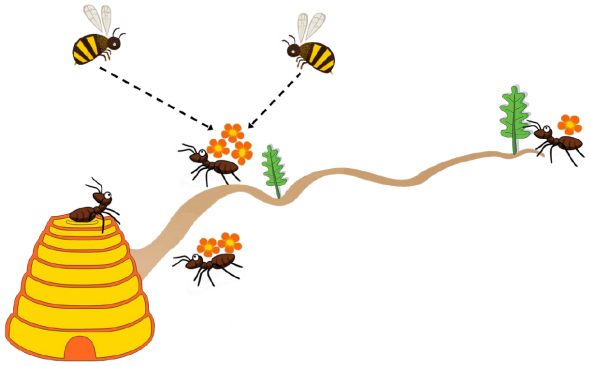

Figure 2. Interaction among the bees and ants of the hierarchic approach (the bees use the best solution obtained by the ants in order to improve it).

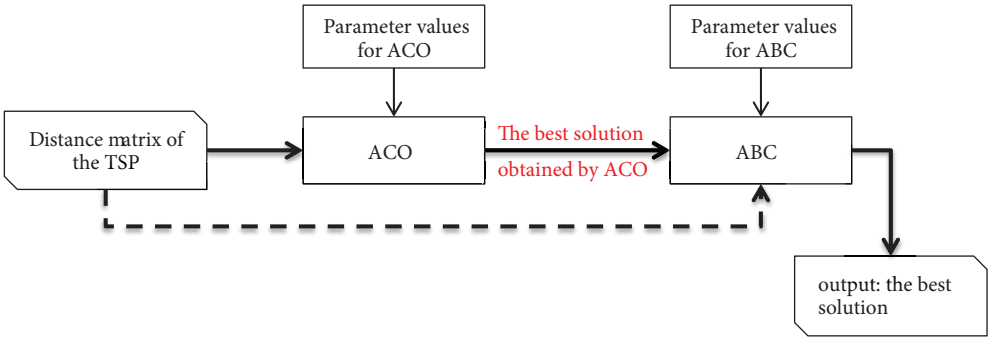

Figure 3. Working diagram of the hierarchic approach.

3. Computational experiments

For all of the experiments, the maximum number of iterations is considered to be 500. For the hierarchic approach, the ACO algorithm is run in the first half of the maximum number of iterations and the ABC is run in the other half. The population size of the algorithms is equal to the number of nodes in the TSP. If the number of nodes of the problem is an odd number, the population size of the ABC is increased by 1, because half of the population is employed bees and the other half is onlooker bees. The optimum tour lengths for the test problems that were obtained from TSPLIB [57] and the parameter settings of the ACO, ABC, and hierarchic approach are given in Tables 4 and 5, respectively. TSPLIB was published in 1991 and is a collection of TSP benchmark instances of varying difficulty; it has been used by many research groups for comparing results. The other TSP instances that are not considered in this study can be found in the TSPLIB.

For the Oliver30, Eil51, Berlin52, St70, Pr76, Eil76, Kroa100, Ch150, and Tsp225 TSPs, ACO, ABC, and the hierarchic approach are conducted 20 times, independently for each problem, and the obtained results are reported as the best, worst, and mean. The relative error (RE) is calculated using Eq. (9) and is displayed in the result tables.

$$RE = \frac{B - O}{O} \times 100 \tag{9}$$

Here O is the optimum tour length of the problem and B is the tour length obtained by the algorithms. For each problem, the results are given in Table 6 in order to clearly compare the methods, and the solutions with the lowest RE obtained by the methods for the problems are given in bold.

Table 4. Number of nodes and optimum tour lengths of the problems.

	Problem	Number of nodes	Optimum tour length
1	Oliver	30	423.74
2	Eil	51	428.87
3	Berlin	52	7544.37
4	St	70	677.11
5	Eil	76	545.39
6	Pr	76	108,159.44
7	Kroa	100	21,285.44
8	Eil	101	642.31
9	Ch	150	6532.28
10	Tsp	225	3859.00

Table 5. Parameter settings of the ABC, ACO, and hierarchic approaches.

Parameters	ABC	ACO	Hierarchic approach	
			ABC	ACO
Population size (P)	D*	D*	D*	D*
Maximum cycle Number	500	500	250	250
Alpha (α)	N/A	1.0	N/A	1
Beta (β)	N/A	5.0	N/A	5.0
Rho (ρ)	N/A	0.65	N/A	0.65
Q	N/A	100	N/A	100
Limit	$(P/2) \times D \times 1000$	N/A	$(P/2) \times D \times 1000$	N/A

*D: Number of nodes in the TSP.

As seen from the results, the hierarchic approach produces better quality solutions than the other methods for all of the test problems and the optimum solutions are also obtained for the Oliver30 and Berlin52 TSPs. In terms of time, the ABC has a shorter running time than ACO and the hierarchic approach, and the proposed approach has a running time of about half that of ACO.

In addition, the proposed method is compared with the RABNET-TSP [58] and oRABNET-TSP [59] solvers. oRABNET-TSP is the original version of the RABNET-TSP solver. The results of RABNET-TSP and oRABNET-TSP for Eil51, Berlin52, Eil76, Eil101, Kroa100, and Ch150 are directly taken from [58]. The comparison of the methods is shown in Table 7.

As seen from Table 7, similar results for the Eil51 and Eil76 problems are obtained by the methods. RABNET-TSP is better than oRABNET-TSP and the hierarchic approach for the Eil101 and Kroa100 test instances. The hierarchic approach is better than RABNET-TSP and oRABNET-TSP for the Berlin52 and Ch150 benchmark problems. Table 7 also shows that the hierarchic approach is an alternative TSP solver and a competitive algorithm.

4. Results and discussion

As seen from the computational experiments (Table 6), the running time of the solution construction-based method (ACO) is long and the running time of the solution improvement-based method (ABC) is short, but the

quality of the solutions obtained by ACO is much better than the solution obtained by ABC, because ACO uses both problem information (distances between nodes) and information of the swarm (pheromone mechanism), and ABC uses only information of the swarm (sharing information in the hive). Figures 4 and 5 give the comparisons of the central processing unit (CPU) time and objective function values of each approach based on the obtained minimum values. As an example, the ABC approach solves all of the problems with minimum CPU times according to the ACO and hierarchic approaches, and for Oliver30, while ACO takes 27.94 times longer to solve it than the ABC, the hierarchic approach takes 15.58 times longer than the ABC. Finally, while ACO consumes maximum CPU times in all of the test problems, the hierarchic and ABC approaches respectively follow it. Figure 5 shows the objective function value comparisons between 3 different approaches based on the optimal values. As seen from Figure 5, the ABC provides the worst objective values, whereas the ACO and hierarchic approaches provide similar and better values. For Tsp225, the ACO, ABC, and hierarchic approaches

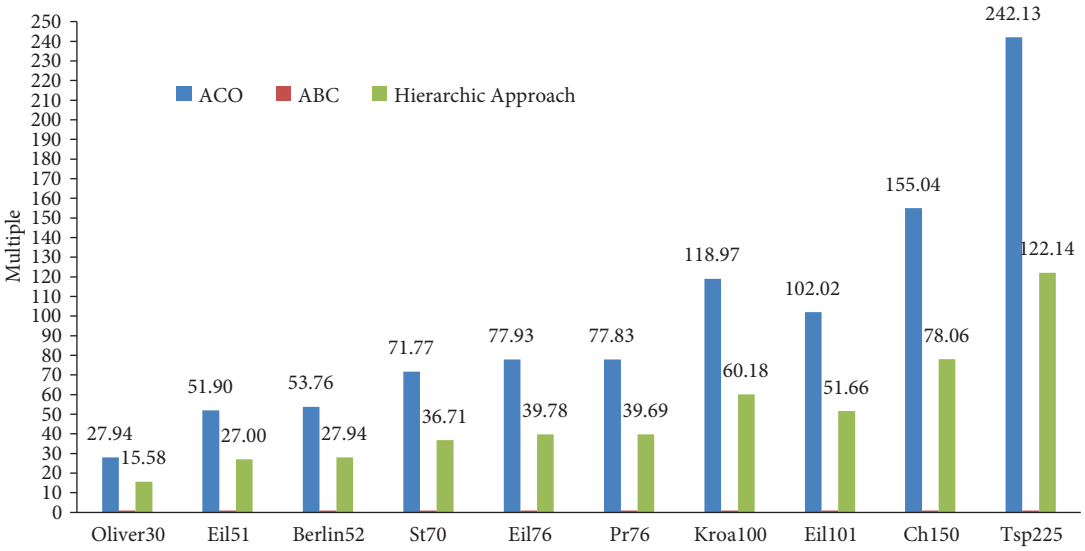

Figure 4. CPU time comparison of the ACO, ABC, and hierarchic approaches.

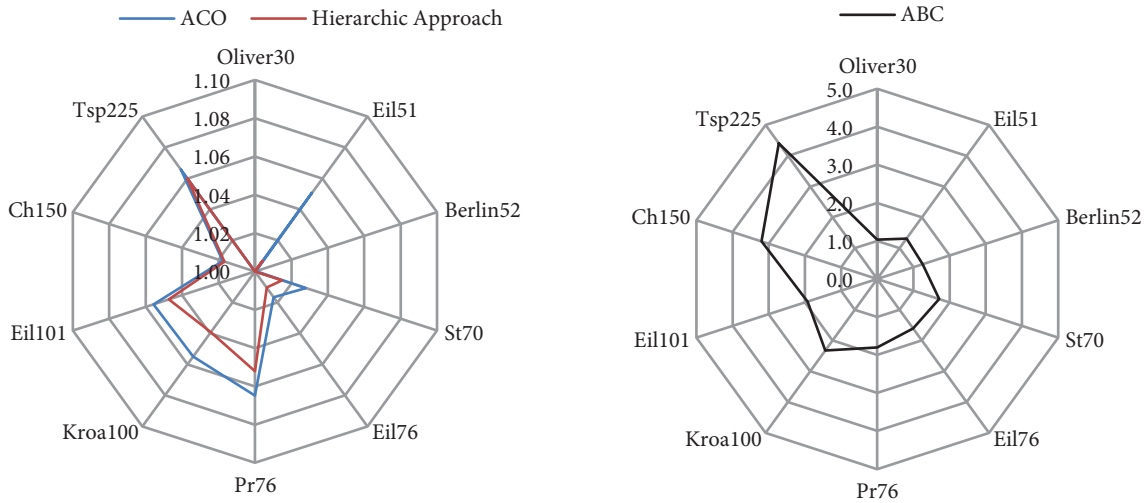

Figure 5. Objective function comparisons of the ACO, ABC, and hierarchic approaches based on optimal values.

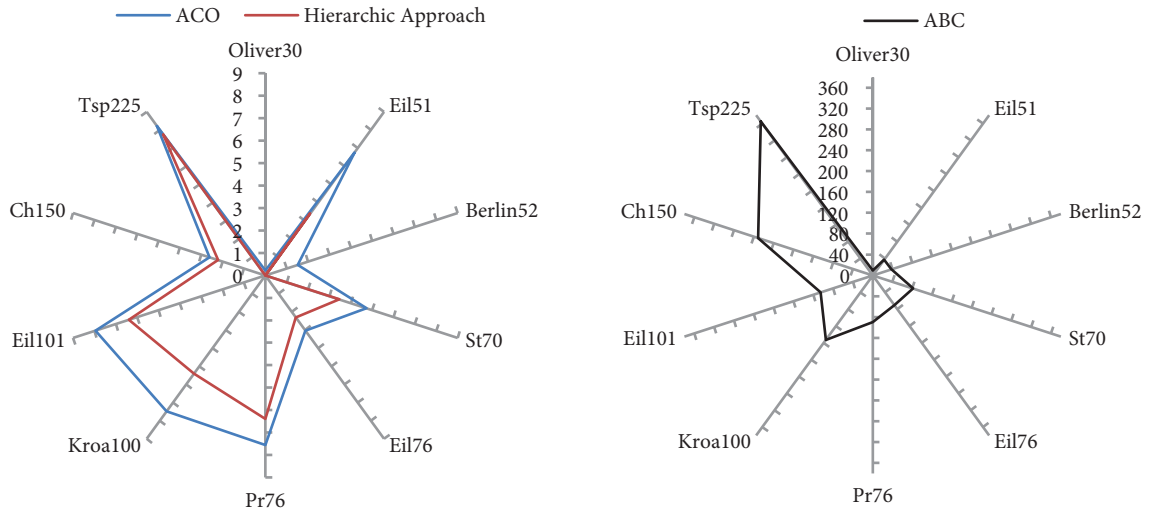

Figure 6. RE (%) comparisons of the ACO, ABC, and hierarchic approaches.

Table 6. Results obtained by the ACO, ABC, and hierarchic approaches for the test problems.

Problem	Method	Best	Worst	Mean	Std. dev.	*REM (%)	Time (s)
Oliver30	ACO	423.74	429.36	424.68	1.41	0.22	35.20
	ABC	439.49	484.83	462.55	12.47	9.16	1.26
	HA**	**423.74**	**423.74**	**423.74**	**0.0**	**0.00**	**19.63**
Eil51	ACO	450.59	463.55	457.86	4.07	6.76	112.11
	ABC	563.75	619.44	590.49	15.79	37.68	2.16
	HA	**431.74**	**454.97**	**443.39**	**5.25**	**3.39**	**58.33**
Berlin52	ACO	7548.99	7681.75	7659.31	38.7	1.52	116.67
	ABC	9479.11	11,021.99	10,390.26	439.69	37.72	2.17
	HA	**7544.37**	**7544.37**	**7544.37**	**0.0**	**0.00**	**60.64**
St70	ACO	696.05	725.26	709.16	8.27	4.73	226.06
	ABC	1162.12	1339.24	1230.49	41.79	81.73	3.15
	HA	**687.24**	**716.52**	**700.58**	**7.51**	**3.47**	**115.65**
Eil76	ACO	554.46	568.62	561.98	3.5	3.04	271.98
	ABC	877.28	971.36	931.44	24.86	70.78	3.49
	HA	**551.07**	**565.51**	**557.98**	**4.1**	**2.31**	**138.82**
Pr76	ACO	115,166.66	118,227.41	116,321.22	885.79	7.55	272.41
	ABC	195,198.9	219,173.64	205,119.61	7379.16	89.65	3.50
	HA	**113,798.56**	**116,353.01**	**115,072.29**	**742.9**	**6.39**	**138.92**
Kroa100	ACO	22,455.89	23,365.46	22,880.12	235.18	7.49	615.06
	ABC	49,519.51	57,566.05	53,840.03	2198.36	152.94	5.17
	HA	**22,122.75**	**23,050.81**	**22,435.31**	**231.34**	**5.40**	**311.12**
Eil101	ACO	678.04	705.65	693.42	6.8	7.96	527.42
	ABC	1237.31	1392.64	1315.95	35.28	104.88	5.17
	HA	**672.71**	**696.04**	**683.39**	**6.56**	**6.39**	**267.08**
Ch150	ACO	6648.51	6726.27	6702.87	20.73	2.61	1387.65
	ABC	20,908.89	22,574.99	21,617.48	453.71	230.93	8.95
	HA	**6641.69**	**6707.86**	**6677.12**	**19.3**	**2.21**	**698.61**
Tsp225	ACO	4112.35	4236.85	4176.08	28.34	8.22	4038.75
	ABC	16,998.41	18,682.56	17,955.12	387.35	365.2792	16.68
	HA	**4090.54**	**4212.08**	**4157.85**	**26.27**	**7.74**	**2037.33**

*: RE relative error of the mean results obtained by 20 runs.

**: Hierarchic approach (HA).

find objective values of 1.066, 4.41, and 1.06 times greater than the optimal value. This ranking is the same for each test problem. The hierarchic approach finds optimal values for Oliver30 and Berlin52; however, the ABC does not find the optimal value for any of the test problems.

Table 7. The comparison of the hierarchic method, RABNET-TSP, and oRABNET-TSP.

	RABNET-TSP [58]			oRABNET-TSP [58]			Hierarchic approach		
Problem	Best	Mean	Std. dev.	Best	Mean	Std. dev.	Best	Mean	Std. dev.
Eil51	**427**	**437.47**	**4.20**	429	438.7	3.52	431.74	443.39	5.25
Berlin52	7542	7932.50	277.25	7716	8073.97	270.14	**7544.37**	**7544.37**	**0**
Eil76	541	556.33	5.30	**542**	**556.1**	**8.03**	551.07	557.98	4.1
Eil101	**638**	**648.63**	**3.85**	641	654.83	6.57	672.71	683.39	6.56
Kroa100	**21,333**	**21,522.73**	**93.34**	21369	21,868.47	245.76	22,122.75	22,435.31	231.34
Ch150	6602	6738.37	76.14	6629	6753.2	83.01	**6641.69**	**6677.12**	**19.3**

We use the methods by combining the advantages of ACO and ABC in order to obtain a better solution in a reasonable time. The results produced by the hierarchic method are better than ACO in terms of time and quality, because ACO shows the stagnation behavior (there is more pheromone on short edges than long edges, and the artificial agents chose short edges with more pheromone) and does not obtain a better solution after a while. Our proposed approach does not show stagnation behavior due to the passing improvement strategy (solution transferring to the ABC). When the hierarchic method is compared with the ABC, the ABC produces a solution in a shorter time, but the solution quality of the hierarchic method is much better than that of the ABC, because the ABC starts to solve the problem randomly but the hierarchic method uses the solution of the ACO at first. Finally, as seen from Figure 6, the obtained REs of the mean values of the hierarchic approach are better than those of ACO and ABC for all of the problems. While the results of the ABC are very far from the optimal values, ACO is better than the ABC; however, it is worse than the proposed hierarchic approach.

5. Conclusion and future works

We propose an effective hierarchic method based on swarm intelligence for solving the problem (ACO) and fast (ABC) algorithms in this study. The numerical tests show that the hierarchic method is an alternative tool for solving the TSP. In future works, we will try to make an alternative pheromone mechanism for ACO, and the better quality solution obtained by ACO will be transferred to methods such as the genetic algorithm, particle swarm optimization, etc., which use the path improvement technique for finding the best solution. Moreover, we will use the proposed method in this paper for solving different optimization problems, such as vehicle routing, supply chain optimization, etc.

Acknowledgments

The authors wish to thank the Selçuk University Scientific Research Project Coordinatorship and the Scientific and Technological Research Council of Turkey (TÜBİTAK) for their institutional support.

References

[1] W.L. Zhong, J. Zhang, W.N. Chen, "A novel discrete particle swarm optimization to solve traveling salesman problem", IEEE Congress on Evolutionary Computation, pp. 3283–3287, 2007.

[2] G. Zweig, "An effective tour construction and improvement procedure for the traveling salesman problem", Operations Research, Vol. 43, pp. 1049–1057, 1995.

[3] A. Uğur, D. Aydın, "An interactive simulation and analysis software for solving TSP using ant colony optimization algorithms", Advances in Engineering Software, Vol. 40, pp. 341–349, 2009.

[4] A. Langevin, F. Soumis, J. Desrosiers, "Classification of travelling salesman problem formulations", Operations Research Letters, Vol. 9, pp. 127–132, 1990.

[5] G. Laporte, "The traveling salesman problem: an overview of exact and approximate algorithms", European Journal of Operational Research, Vol. 59, pp. 231–247, 1992.

[6] A.P. Punnen, "Travelling salesman problem under categorization", Operations Research Letters, Vol. 12, pp. 89–95, 1992.

[7] T. Bektaş, "The multiple traveling salesman problems: an overview of formulations and solution procedures", Omega, Vol. 34, pp. 209–219, 2006.

[8] C. Rego, D. Gamboa, F. Glover, C. Osterman, "Traveling salesman problem heuristics: leading methods, implementations and latest advances", European Journal of Operational Research, Vol. 211, pp. 427–441, 2011.

[9] E.L. Lawler, J.K. Lenstra, A.H.G.R. Kan, D.B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, New York, NY, USA, Wiley, 1985.

[10] G. Gutin, A.P. Punnen, The Traveling Salesman Problem and its Variations, Dordrecht, the Netherlands, Kluwer, 2002.

[11] D.L. Applegate, R.E. Bixby, V. Chvatal, W.J. Cook, The Traveling Salesman Problem: A Computational Study, Princeton, NJ, USA, Princeton University Press, 2006.

[12] R. Radharamanan R, L.I. Choi, "A branch and bound algorithm for the travelling salesman and the transportation routing problems", Computers & Industrial Engineering, Vol. 11, pp. 236–240, 1986.

[13] K.N. Singh, D.L. Oudheusden, "A branch and bound algorithm for the traveling purchaser problem", European Journal of Operational Research, Vol. 97, pp. 571–579, 1997.

[14] B. Fleischmann, "A cutting plane procedure for the traveling salesman problem on road networks", European Journal of Operational Research, Vol. 21, pp. 307–317, 1985.

[15] M. Padberg, G. Rinaldi, "Optimization of a 532-city symmetric traveling salesman problem by branch and cut", Operations Research Letters, Vol. 6, pp. 1–7, 1987.

[16] H.H. Perez, J.J.S. Gonzalez, "A branch and cut algorithm for a traveling salesman problem with pickup and delivery", Discrete Applied Mathematics, Vol. 145, pp. 126–139, 2004.

[17] A.G. Chentsov, L.N. Korotayeva, "The dynamic programming method in the generalized traveling salesman problem", Mathematical and Computer Modeling, Vol. 25, pp. 93–105, 1997.

[18] Ö. Ergan, J.B. Orlin, "A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem", Discrete Optimization, Vol. 3, pp. 78–85, 2006.

[19] J.J. Grefenstette, R. Gopal, B. Rosimaita, D.V. Gucht, "Genetic algorithms for the traveling salesman problem", Proceedings of the International Conference on Genetic Algorithms and their Applications, pp. 160–168, 1997.

[20] P. Jog, J.Y. Suh, D.V. Gucht, "The effect of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem", Proceedings of the 3rd International Conference on Genetic Algorithms and their Applications, pp. 110–115, 1989.

[21] L. Qu, R. Sun, "A synergetic approach to genetic algorithms for solving traveling salesman problem", Information Sciences, Vol. 117, pp. 267–283, 1999.

[22] P. Larranaga, C. Kuijpers, R. Murga, I. Inza, S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: a review of representations and operators", Artificial Intelligence Review, Vol. 13, pp. 129–170, 1999.

[23] S.S. Ray, S. Bandyopadhyay, S.K. Pal, "New operators of genetic algorithms for travelling salesman problem", Proceedings of the 17th International Conference on Pattern Recognition, pp. 497–500, 2004.

[24] S.B. Liu, K.M. Ng, H.L. Ong, "A new heuristic algorithm for the classical symmetric traveling salesman problem", International Journal of Mathematical Science, Vol. 1, pp. 234–238, 2007.

[25] J. Yang, C. Wu, H.P. Lee, Y. Liang, "Solving travelling salesman problems using generalized chromosome genetic algorithm", Progress in Natural Science, Vol. 18, pp. 887–892, 2008.

[26] J. Majumdar, A.K. Bhunia, "Genetic algorithm for asymmetric traveling salesman problem with imprecise travel times", Journal of Computational and Applied Mathematics Vol. 235, pp. 3063–3078, 2011.

[27] J. Knox, "Tabu search performance on the symmetric traveling salesman problem", Computers & Operations Research, Vol. 21, pp. 867–876, 1994.

[28] M. Gendreau, G. Laporte, F. Semet, "A tabu search heuristic for the undirected selective travelling salesman problem", European Journal of Operational Research, Vol. 106, pp. 539–545, 1998.

[29] J.R.A. Allwright, D.B. Carpenter, "A distributed implementation of simulated annealing for the traveling salesman problem", Parallel Computing, Vol. 10, 335–338, 1989.

[30] X. Geng, Z. Chen, W. Yang, D. Shi, K. Zhao, "Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search", Applied Soft Computing, Vol. 11, 3680–3689, 2011.

[31] H. Ghaziri, I.H. Osman, "A neural network algorithm for the traveling salesman problem with backhauls", Computers & Industrial Engineering, Vol. 44, pp. 267–281, 2003.

[32] K.S. Leung, H.D. Jin, Z.B. Xu, "An expanding self-organizing neural network for the traveling salesman problem", Neurocomputing, Vol. 62, pp. 267–292, 2004.

[33] W. Pang, K.P. Wang, C.G. Zhou, L.J. Dong, M. Liu, H.Y. Zhang, J.Y. Wang, "Modified particle swarm optimization based on space transformation for solving traveling salesman problem", Proceedings of the 33rd International Conference on Machine Learning and Cybernetics, pp. 2342–2346, 2004.

[34] W. Pang, K.P. Wang, C.G. Zhou, L.J. Dong, "Fuzzy discrete particle swarm optimization for solving traveling salesman problem", Proceedings of the 4th International Conference on Computer and Information Technology, pp. 796–800, 2004.

[35] C. Wang, J. Zhang, J. Yang, C. Hu, J. Liu, "A modified particle swarm optimization algorithm and its application for solving traveling salesman problem", International Conference on Neural Networks and Brain, Vol. 2, pp. 689–694, 2005.

[36] 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, Vol. 103, pp. 169–176, 2007.

[37] M. Dorigo, L.M. Gambardella, "Ant colonies for the travelling salesman problem", BioSystems, Vol. 43, pp. 73–81, 1997.

[38] C.F. Tsai, C.W. Tsai, C.C. Tseng, "A new hybrid heuristic approach for solving large traveling salesman problem", Information Sciences, Vol. 166, 67–81, 2004.

[39] A. Puris, R. Bello, Y. Martinez, A. Nowe, "Two-stage ant colony optimization for solving the traveling salesman problem", Lecture Notes in Computer Science, Vol. 4528, pp. 307–316, 2007.

[40] B. Bontoux, D. Feillet, "Ant colony optimization for the traveling purchaser problem", Computers & Operations Research, Vol. 35, pp. 628–637, 2008.

[41] A. Puris, R. Bello, F. Herrera, "Analysis of the efficacy of a two-stage methodology for ant colony optimization: case of study with TSP and QAP", Expert Systems with Applications, Vol. 37, pp. 5443–5453, 2010.

[42] L.P. Wong, M.Y.H. Low, C.S. Chong, "A bee colony optimization algorithm for traveling salesman problem", Proceedings of the 2nd Asia International Conference on Modeling and Simulation, pp. 818–823, 2008.

[43] Y. Marinakis, M. Marinaki, G. Dounias, "Honey bees mating optimization algorithm for the Euclidean traveling salesman problem", Information Sciences, Vol. 181, pp. 4684–4698, 2011.

[44] D. Karaboğa, B.A. Görkemli, "A combinatorial artificial bee colony algorithm for traveling salesman problem", International Symposium on Innovations in Intelligent Systems and Applications, pp. 50–53, 2011.

[45] W.H. Li, W.J. Li, Y. Yang, H.Q. Liao, J.L. Li, X.P. Zheng, "Artificial bee colony algorithm for traveling salesman problem", Advanced Materials Research, Vol. 314, pp. 2191–2196, 2011.

[46] L. Fang, P. Chen, S. Liu, "Particle swarm optimization with simulated annealing for TSP", Proceedings of the 6th WSEAS International Conference on Artificial Intelligence, pp. 206–210, 2007.

[47] R. Takahashi, "A hybrid method of genetic algorithms and ant colony optimization to solve the traveling salesman problem", International Conference on Machine Learning and Applications, pp. 81–88, 2009.

[48] D. Gomez-Cabrero, C. Armero, D.N. Ranasinghe, "The traveling salesman's problem: a self-adapting PSO-ACS algorithm", Second International Conference on Industrial and Information Systems, pp. 479–484, 2007.

[49] H.K. Feng, J.S. Bao, Y. Jin, "Particle swarm optimization combined with ant colony optimization for the multiple traveling salesman problems", Materials Science Forum, Vol. 626–627, pp. 717–722, 2009.

[50] S.M. Chen, C.Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques", Expert Systems with Applications, Vol. 38, pp. 14439–14450, 2011.

[51] M. Dorigo, V. Maniezzo, A. Colorni, "Ant system: optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, Vol. 26, pp. 29–41, 1996.

[52] D. Karaboğa, "An idea based on honeybee swarm for numerical optimization", Technical Report-TR06, Kayseri, Turkey, Erciyes University, 2005.

[53] W.Y. Szeto, Y. Wu, S.C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem", European Journal of Operational Research, Vol. 215, pp. 126–135, 2011.

[54] Q.K. Pan, M.F. Tasgetiren, P.N. Suganthan, T.J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem", Information Sciences, Vol. 181, pp. 2455–2468, 2011.

[55] M.S. Kıran, H. İşcan, M. Gündüz, "The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem", Neural Computing and Applications, Vol. 23, pp. 9–21, 2013.

[56] D. Aydın, "Adaptation of swarm intelligence approaches into color image segmentation and their implementations on recognition systems", PhD, Ege University, İzmir, Turkey, 2011.

[57] G. Reinelt, TSPLIB, Traveling Salesman Problem Benchmark Problems, Heidelberg, Germany, Heidelberg University, available at http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[58] 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, Vol. 179, pp. 1454–1468, 2009.

[59] R. Pasti, L.N. de Castro, "A neuro-immune network for solving the traveling salesman problem", Proceedings of International Joint Conference on Neural Networks, pp. 3760–3766, 2006.