Optimizing The Tour On 3D Cuboid Structures With Particle Swarm Optimization Method

[1]Hüseyin Eldem and *[2]Erkan Ülker
*[1] Karamanoğlu Mehmetbey University, Computer Technologies Department, Karaman, Turkey
[2] Selçuk University, Computer Engineering Department, Campus, Konya, Turkey

Abstract

Traveling Salesman Problem (TSP) is real world problem of finding the shortest (minimum cost) possible routes that visits each node in a given set of nodes (cities) once and then returns to origin city. The optimization of cuboids has potential samples that can be adapted to real world. For example rooms, buildings, furniture etc. Different optimization algorithms have been used in solution of optimization problems at present. Among them, meta-heuristic algorithms come first. In this study, particle swarm optimization, one of meta-heuristic methods, is applied for solving Euclidian TSP consisting of nine different sized sets of nodes randomly placed on a cuboid surface. Method performance is corroborated with tests.

Key words: Particle Swarm Optimization, Cuboid Structures, Metaheuristic, Euclidean *TSP*, Path-relinking, Path planning

1. Introduction

Traveling salesman problem (*TSP*) is a well-known problem, which has been widely studied in the engineering applications and computer sciences. TSP can be defined that has to be solved by a salesman who travels between cities at minimum cost and return to the origin city. In this problem, one of the parameters including cost, time and path is optimized. The problem may be called as a Hamiltonian cycle problem investigated in the scope of a graph theory. In similar popular problems these are discrete and combinatorial, TSP also has been widely studied. If all costs between two cities are equal for both cities ($d_{ij} = d_{ji}$), *TSP* is named as symmetric, and otherwise, it is called asymmetric *TSP* ($d_{ij} \neq d_{ji}$). Two-dimensional Euclidean *TSP* is a popular and widely studied issue [1]. Developing effective *TSP* solutions is gradually gaining importance in route planning problems such as airways, delivery trucks, mail carriers and computer networks.

For to find best solutions on TSP many methods have been developed. These are grouped by exact methods and heuristic techniques. Iterative improvement, branch-and-bound and branch-and-cut methots are exact for TSP [2-3]. Some heuristic techniques are based on Simulated Annealing [4],Genetic Algorithms (GA) [5-7], Tabu Search [8-9], Artificial Neural Networks [10-12] and Ant Colony Systems [13-18] have been developed for solving *TSPs*. In some researchs to attain better results for *TSPs*, hybrid evolutionary algorithms have been proposed [18-21]. In [22-25], *TSP* applications were performed for the points on 3D geometric shapes like

*Corresponding author: Address: Technical Vocational High School, Department of Computer Technologies Karamanoğlu Mehmetbey University, 70100, Karaman TURKEY. E-mail address: heldem@kmu.edu.tr, Phone: +903382262177 Fax: +903382262166

sphere and cuboid. Algorithms were developed through using genetic algorithms for *TSP* solution on sphere [22] on a cuboid [23]. For cuboid shapes also has solved TSP by *PSO* [24] and *ACO* [24].

Particle Swarm Optimization (*PSO*) is one of metaheuristic algorithms used to solve optimization problems, was proposed by as a Dr. Eberhart ve Dr. Kennedy in 1995 [26]. PSO is an population based optimization technique, evaluated in the swarm intelligence category. Like people talking or sharing information among them, behaviour of bird flocking or fish schooling asserts a social intelligence between them. *PSO*, inspired from social behavior of fish and birds. Members in population of PSO are called as "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles [26].

In this study, *TSP*, that is optimization of described points on a cuboid shape is solved by *PSO* algorithm. For the existing *TSP*s the coordinates of the points or the distances between them are known. The problem, subject to this study is different from the existing *TSP*s. Because all points are located on a cuboid shape and the transition between points can be made through cuboid surface. Definitions of points on the cuboid surface and finding distance between all points among them are explained. After this, the adaptation of the problem to PSO is explained in the further parts of this study. In real life, the solution could be used on the surface of object or structure for potential searching and rescue action to optimize each criteria of problem. For example, when a fire in a skyscraper and can't reach to floors and rooms, rescue helicopter move over the surface of building. And this helicopter must be complete this job immediately. In this example, to complete the rescue by helicopter on the surface of building should be optimized.

In this study, the performance of the developed method using *PSO* is tested, also its comparison with *GA* selected from the literature. Primarily, the definitions of points on cuboid surface and determination of distances between points are explained. Subsequently, the adaptation of the problem to *PSO* is explained in the further parts of the paper.

2. Materials and Method

2.1. Definition Of Cuboid Shapes with Mathematical Notation

Cuboid is geometric shape which can be frequently seen in various products including wooden, metal, plastic and paper boxes, books, toys, dice, cupboards, furniture, rooms and buildings.

Cuboid is a three-dimensional object and a geometric shape with six rectangular faces (Fig. 1). It has six faces, twelve sides and eight corners. Among geometric shapes, rectangular prism has a cuboid shape. Among three-dimensional objects, cube is a special form of cuboid, consisting of all square faces. As can be seen in Fig. 1, cuboid is one of the three-dimensional objects with height, width and length.

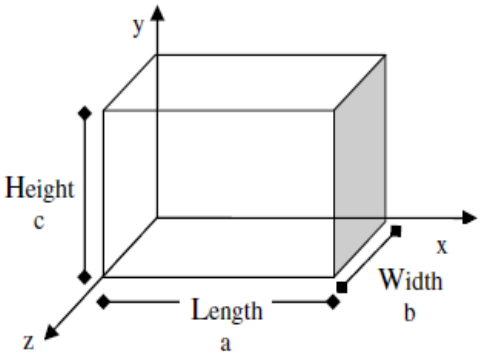


Figure 1. A Cuboid Shape

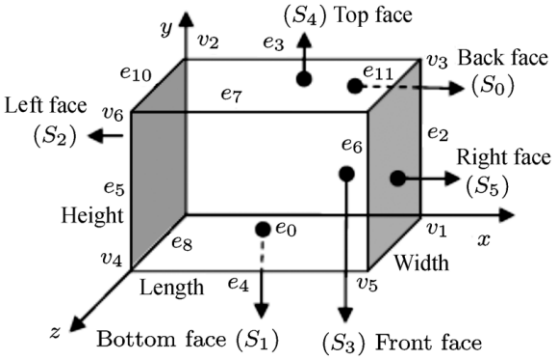


Figure 1. A Cuboid Object

Cuboid objects have six faces: back (S_0), bottom (S_1), left (S_2), front (S_3), top (S_4) and right (S_5) (Fig. 2). Geometric information about a cuboid object located as back face z=0, bottom y=0 and left x=0 is given in Table 1. There are four sides and four corners for each face and some sides and corners are shared among the rectangles forming the face (Table 1).

Table 1 Geometrical data of a cuboid

Faces	x	y	z	Vertices	Edges
S0 (Back Face)	$0 < x < Length$	$0 < y < Height$	0	0,1,3,2	0,2,3,1
S1 (Bottom Face)	$0 < x < Length$	0	$0 < z < Width$	0,4,5,1	0,8,4,9
S2 (Left Face)	0	$0 < y < Height$	$0 < z < Width$	0,2,6,4	1,10,5,8
S3 (Front Face)	$0 < x < Length$	$0 < y < Height$	$Width$	4,5,7,6	4,6,7,5
S4 (Top Face)	$0 < x < Length$	$Height$	$0 < z < Width$	2,3,7,6	3,11,7,10
S5 (Right Face)	$Length$	$0 < y < Height$	$0 < z < Width$	1,3,7,5	2,11,6,9

N points could be present on any of 6 cuboid faces. *N*=7 cities ($c_0, c_1, c_2, c_3, c_4, c_5$ *and* c_6) could be located on cuboid faces ($S_3, S_1, S_5, S_4, S_4, S_0$ *and* S_2) as in Fig. 3.

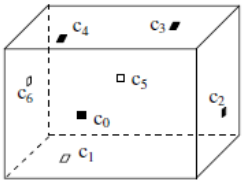


Figure 2. View of Seven Points on Cuboid Faces

The Euclidean distance of a pair of cities c_i (x_i, y_i, z_i) and c_j (x_j, y_j, z_j), is notated as $d_{ij}, (i, j = 1 \ldots . n)$. Then the distances between all pairs of cities can be computed in an *N*x*N* symmetric distance matrix *D*=[d_{ij}]. For a pair of cities on the cuboid, there are three possibilities:

(a) both on the same face; (b) two cities on opposite faces; (c) two cities on adjacent faces. The pseudocode used for calculating the distance between c_i and c_j points is given in Table 2.

Table 2. Pseudocode to calculate the distance between two points

if (face(c_i) == face(c_j)) //SAME FACE
    d_{ij} = Euclidean distance between the two points;
else if (face(c_i) (opposite) face(c_j)) //OPPOSITE FACE
{
    calculate four alternative path distances d_i (through remaining faces)
    $d_{ij} = d_0$;
    for(int k = 0; k < 4; ++k)
    if($d_k < d_{ij}$) $d_{ij} = d_k$;
}
else if (face(c_i) (adjacent) face(c_j)) //ADJACENT FACE
{
    d_0 = calculate distance between the two points (through common edge of faces containing c_1 and c_2);
    d_1 = calculate distance between the two points (through the first common adjacent face of c_1 and c_2);
    d_2 = calculate distance between the two points (through the second common adjacent face of c_1 and c_2);
    d_{ij} = min(d_0, d_1, d_2);
}

The detailed information about pseudocode above used for calculating the distance between two points on cuboid faces is as follows:

(a) **If two cities are on the same face:** Euclidean distance between two points is computed directly by $\sqrt{(x_j - x_i)^2+(y_j - y_i)^2+(z_j - z_i)^2}$. The combinations are $\{S_0, S_0\}$, $\{S_1, S_1\}$,$\{S_2, S_2\}$,$\{S_3, S_3\}$,$\{S_4, S_4\}$ and $\{S_5, S_5\}$ for any city pairs c_i and c_j.

(b) **If two cities are on the opposite faces:** There are three alternatives in this case. (i) Front-Back $\{S_3, S_0\}$ (ii) Left-Right $\{S_2, S_5\}$ and (iii) Bottom-Top $\{S_1, S_4\}$. Four possible routes must be calculated for measuring the distance between opposite faces and the shortest path must be chosen. For instance, the possible routes for Left- Right case are, (I) Left-Front-Right, (II) Left-Back-Right, (III) Left-Top-Right and (IV) Left-Bottom-Right.

In this case, the possible routes (d_1, d_2, d_3 and d_4) can be calculated, respectively and then, the shortest route is chosen. For example calculation of d_1, d_2 and d_3 is given below.

$$for\ alternative\ (I) \rightarrow \left\{ \begin{array}{l} d_y = \ y_j - y_i \\ d_z = (width - \ z_j) + \ length + \ (width - \ z_i) \\ d_1 = \sqrt{{d_y}^2 + \ {d_z}^2} \end{array} \right\}$$

$$for\ alternative\ (II) \rightarrow \begin{cases} d_y = \ y_j - y_i \\ d_z = z_j + \ length + \ z_j \\ d_2 = \sqrt{{d_y}^2 + \ {d_z}^2} \end{cases}$$

$$for\ alternative\ (III) \rightarrow \begin{cases} d_y = \ (height - \ y_j) + \ length + \ (height - \ y_i) \\ d_z = \ z_j - z_i \\ d_3 = \sqrt{{d_y}^2 + \ {d_z}^2} \end{cases}$$

The routes can be similarly calculated for the other two cases mentioned above.

(c) **If two cities are on neighboring (adjacent) faces:** There are again four alternative routes between adjacent faces in this case. However, if another face is visited for calculating the distance between adjacent faces, this alternative is neglected in calculations as it is the longest. For instance, for Front-Bottom neighboring faces, (i) Front-Bottom, (ii) Front-Right-Bottom, (iii) Front-Left-Bottom and (iv) Front-Top-Back-Bottom alternatives must be calculated. However, as the nonneighboring Top and Back faces are present for Front-Bottom transition in (iv) Front-Top-Bottom alternative, it is not taken into consideration and neglected. In this case, the possible alternative routes ($d_1, d_2\ and\ d_3$) are calculated, respectively, and then, the shortest route is chosen. For example calculation of $d_1\ and\ d_2$is given below.

$$for\ alternative\ (I) \rightarrow \begin{cases} d_x = \ x_j - x_i \\ d_z = (width - \ z_j) + y_i \\ d_1 = \sqrt{{d_x}^2 + \ {d_z}^2} \end{cases}$$

$$for\ alternative\ (II) \rightarrow \begin{cases} d_x = \ x_j + y_i \\ d_z = (width - \ z_j) + x_i \\ d_2 = \sqrt{{d_x}^2 + \ {d_z}^2} \end{cases}$$

There are 12 alternatives as neighboring faces: (Back-Bottom), (Back-Left), (Back-Top), (Back-Right), (Bottom-Left), (Bottom-Front), (Bottom-Right), (Left-Front), (Left-Top), (Front-Top), (Front-Right) and (Top-Right). The distance of each combination can be computed in a similar way by considering axis differences.

2.2. Solution of Cuboid Surface TSP By Using PSO

3D TSP that would be applied to the surface of the cuboid is different from normal *2D TSP* [23,25]. The salesman could only travel between points located on the surface of the cuboid. The only difference in this problem is that the points are not located in the cuboid but on the surface of the cuboid.

The problem to be solved can be defined as the determination of the minimum tour distance for a salesman (ant) to travel all points (*N* points in total with known coordinates) located on a surface of a cuboid and come back to the original point, similar to standard *TSP*. In this study, it is aimed to solve the depicted problem by *PSO* method.

After creating a distance matrix consisting of each pair of points, problem solution becomes the same as standard *TSP*. *PSO* algorithm is adapted for use in the *TSP*s solution was given in Table 3.

Table 3. General Structure of PSO adapted to the problem.

```
FOR each particle
        Initialize particles
END

DO
    FOR each particle
        Calculate fitness value (tour length)
        IF fitness value (tour length) better than pbest ,
            Set current tour length as the new pbest
      END

 Set as gbest for each particle the best pbests of all particle  (minimum tour length)

    FOR each particle
        Calculate particle velocities
        Update particle positions
    END
WHILE maximum iterations or minimum error criteria is not attained
```

According to this general structure (Table 3), firstly initial particles of PSO algorithm randomly generated . After, distance matris consisting of distances between each point and every other point is created. Fitness values of each particle are calculated by distance matrix. Particle it has minimum tour length is determined as global best. By finding pbest and gbest, the following Eq.(1) for velocity and Eq.(2) for position is used. For each iteration, if the particles find shortest tour length before, new value replaced with previous. After, new gbest is determined again.

$$v_i^{k+1} = w.v_i^k + c_1.rand_1^k.\left(pbest_i^k - x_i^k\right) + c_2.rand_2^k.\left(gbest^k - x_i^k\right) \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (2)$$

Where, ***rand*** between 0 and 1 a generated random value, ***i*** is particle number, ***k*** is number of iteration. c_1 and c_2 are learning factors (weights). c_1 directs particles according to their own experiences (personal memory), c_2 directs particles according to other particles experience (social

memory). Usually if c_1 and c_2 set to 2 value, results of algorithm are good. w is inertia weight that is reduce the impact of velocity progressively for each iteration.

3. Experimental Results

The simulation results obtained through this software were produced for the points N = 10, 20, 30, 40, 50, 100, 150, 200, 250 on a 1000x1000x1000 cuboid. Simulations were repeated 100-times for each value of N. A new random point set was generated for each trial. In order to generalize results predefined points set not used, randomly generate points set preffered. Results was performed using Matlab R2010 programming language. The performance of the proposed method were compared with the results by the proposed GA [23].

Results, representing the optimal tour length was obtained in 20, 40, 60, 80, 100 evolution. For all experiments, the learning factor c_1 and c_2 was taken as 2. Results are obtained for the lines on the surface of the cuboid not in the cuboid. For both the proposed PSO and GA approaches for a specified number of points average tour lengths after the 100 iteration are presented are given in Table 4. In Fig. 5, the results obtained are given in diagram form.

The optimum route determined for 100, 250 and 400 points by *cuboidTSP* is given in Fig. 4. All points and route can be simultaneously seen in transparent mode.

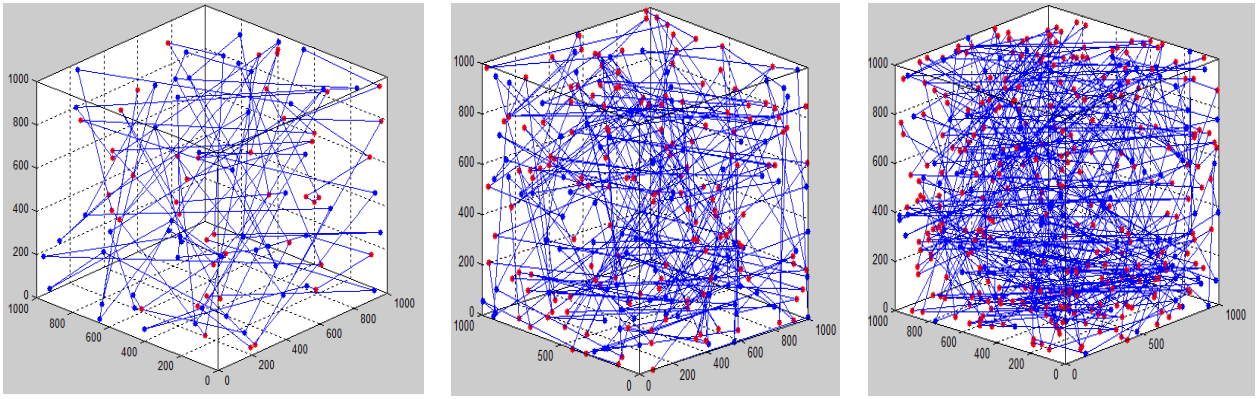
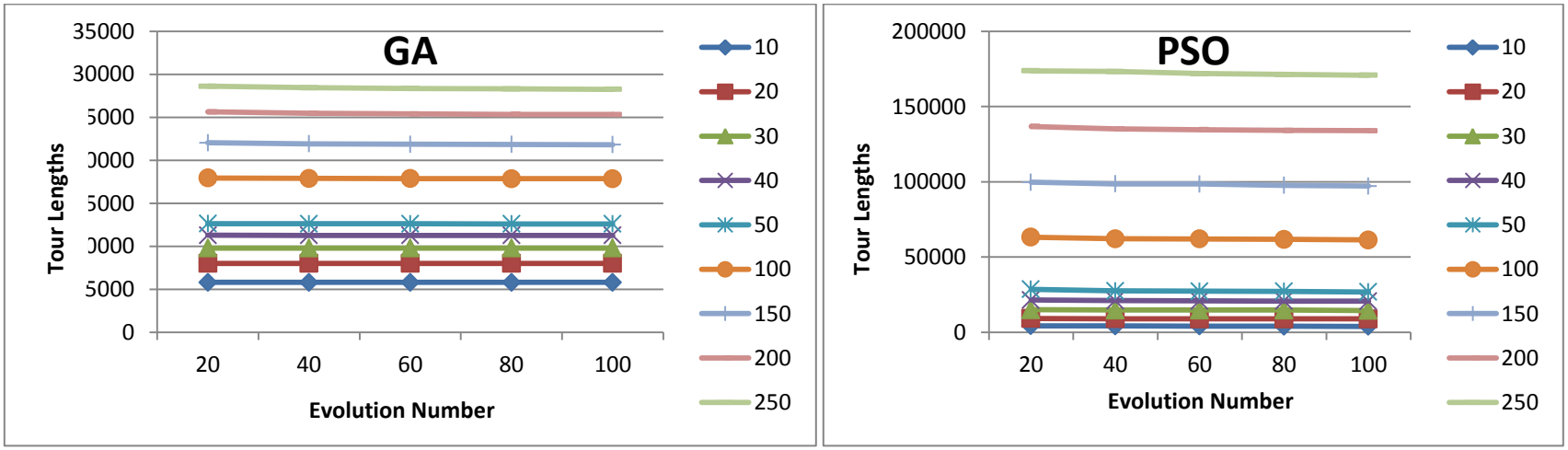


Figure 4. The transparent view of the shortest routes obtained for 100, 250 and 400 points randomly placed on the cuboid.

Through increasing the numbers of tour and evolution in algorithm, the optimum results can be further improved, as can be seen in Table 4 and Fig. 5. By altering the *PSO* parameters, better results can be obtained. *PSO* results can also be improved through using available heuristic and hybrid methods for *TSP* solution.

Table 4. Calculated average tour lengths with *GA* [23] and *PSO* for *N* = 10, 20, 30, 40, 50, 100, 150, 200, 250 points on the surface of the cuboid.

Evolution Number	Number of Points																	
	10		20		30		40		50		100		150		200		250	
	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO	GA	PSO
20	5805.182	4268,6716	8002.746	9179,5164	9783.089	14882,499	11269.02	21409,904	12631.87	28479,609	17938.38	63140,068	22034.3	99617,853	25623.9	136757,04	28612.86	173742,82
40	5805.182	4133,1715	8002.746	8993,7784	9782.443	14741,698	11264.42	20945,987	12614.53	27440,989	17894.14	62040,927	21902.88	98547,888	25457.59	135079,66	28430.73	173274,67
60	5805.182	4051,5134	8002.746	8902,3228	9782.443	14731,523	11263.1	20837,005	12609.98	27209,321	17874.84	61995,757	21864.79	98509,966	25383.34	134551,58	28347.62	171889,71
80	5805.182	3998,9603	8002.746	8878,1357	9782.443	14685,058	11261.98	20660,114	12602.5	27176,016	17853.17	61654,65	21833.4	97628,667	25341.58	134079,88	28293.63	171266,39
100	5805.182	3860,5593	8002.746	8862,3634	9782.443	14426,52	11261.76	20628,726	12598.33	26764,498	17845.66	61269,237	21813.08	97085,37	25317.06	133912,15	28252	170750,14



(a) (b)

Figure 5. Average tour lengths for different amount points on the surfaces of the cuboid founded bys *GA* [23] and *PSO* solutions.

Conclusions

The main contribution of this paper is to show the applicability of *TSP* solutions cuboid objects and structures of any size. It is quite important for saving from time in route planning of air vehicles like helicopter in search and rescue operations on cuboid structures during natural disasters like fire or earthquake. Furthermore, the succesful optimization of the method could be used in maintenance (painting etc.) or cleaning operations on cuboid structures. This method can be used in any kind of applications requiring the use of programmable wall climbing robots. It is possible that *3D* applications of *TSP* problems will become more important with the advancement of robotic industry in the future.

The current optimization techniques in literature have been generally developed in *2D* environments. Developing and using these techniques in *3D* environments will definitely inspire the development of optimization techniques in different fields.

In the future studies, the other methods in the solution of *TSP*, e.g. *PSO*, can be tested for the solution of the *cuboidTSP*. Meanwhile, *cuboidTSP* problems can be studied by hybrid utilization of *ACO* and *PSO* methods.

References

[1] Arora S., Polynomial time approximation schemes for Euclidean TSP and other geometric problems, Proc. 37th Ann. Symp. Foundations of Computer Sci., IEEE Computer Soc., 1996; 2-11.

[2] Dantzig G., Fulkerson R., Johnson S., Solution of a Large-Scale Travelling Salesman Problem, Journal of Operations Research Society. 1954;2:393-410.

[3] Laporte G., "The Vehicle Routing Problem: An overview of exact and approximate algorithms", European Journal of Operational Research, 1992;59:345-358.

[4] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P.: Optimization by Simulated Annealing. Science, 1983;220:671-680.

[5] Tsujimura Y., Gen M., Entropy-based genetic algorithm for solving TSP, Knowledge-Based Intelligent Electronic Systems 1998;2:285-290.

[6] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Reading; 1989.

[7] Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor; 1975.

[8] Glover, F.: Tabu Search - Part I. ORSA Journal on Computing 1 1989;3:190-206.

[9] Glover, F.: Tabu Search - Part II. ORSA Journal on Computing 2 1990;1:4-32.

[10] Hopfield, J.J. and Tank, D.W.: Neural Computation of Decisions in Optimization Problems. Biological Cybernetics 1985;52:141-152.

[11] Kohonen, T.: Self-Organizing Maps. 3rd ed. Springer-Verlag Berlin 2001.
[12] Shinozawa K., Uchiyama T. and Shimohara K., An approach for solving dynamic TSPs using neural networks, Neural Networks, IEEE International Joint Conference 1991;3:2450-2454.

[13] Colorni, A., Dorigo, M. and Maniezzo, V.: Distributed Optimization by Ant Colonies. In Proceedings of the European Conference on Artificial Life (ECAL'91, Paris, France), eds F. Varela and P. Bourgine. Elsevier Publishing, Amsterdam, 1991;134-142.
[14] Colorni, A., Dorigo, M. and Maniezzo, V.: An investigation of some properties of an ant algorithm. In Proceedings of the Second Conference on Parallel Problem Solving from Nature (PPSN II, Brussels, Belgium), eds R. Maenner and B. Manderick. North-Holland, Amsterdam, 1992; 509-520.
[15] Gambardella L.M. and Dorigo M., Solving Symmetric and Asymmetric TSPs by Ant Colonies, International Conference on Evolutionary Computation, Nagoya, Japan 1996; 622-627.
[16] Dorigo, M. and Gambardella, L.M., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE Transactıons On Evolutıonary Computation, 1997a;1:53-66.
[17] Dorigo, M. and Gambardella, L. M., "Ant Colonies for the Travelling Salesman Problem", BioSystems 1997b;43:73-81.
[18] Lee Z. J., A hybrid algorithm applied to travelling salesman problem, Networking, Sensing and Control, IEEE International Conference 2004;1:237-242.
[19] White C. M. and Yen G. G., A hybrid evolutionary algorithm for traveling salesman problem, Congress on Evolutionary Computation (CEC2004) 2004;2:1473-1478.
[20] Marinakis Y., Migdalas A. and Pardalos P. M., A Hybrid Genetic-GRASP Algorithm Using Lagrangean Relaxation for the Traveling Salesman Problem, J. Comb. Optim. 2005;10:311-326.
[21] Tsai, C., Tsai, C. and Tseng, C., "A New Hybrid Heuristic Approach for Solving Large Traveling Salesman Problem", Information Sciences, 2004; 166: 67-81.
[22] Uğur A., Korukoğlu S., Calıskan A., Cinsdikici M., Alp A., Genetic Algorithm Based Solution For Tsp On A Sphere, Mathematical and Comp. Applications 2009;14(3):219-228.
[23] Uğur A., Path Planning On A Cuboid Using Genetic Algorithms, Information Sciences 2008;178:3275-3287.
[24] Shoubao S. and Xibin C., Jumping PSO with Expanding Neighborhood Search for TSP on a Cuboid, Chinese Journal of Electronics, 2013;22 (1):202-208.

[25] Eldem H. and Ülker E., Application Of Ant Colony Optimization For The Solution Of 3 Dimensional Cuboid Structures, Journal of Computer and Communications, 2014;2(4): 99-107.
[26] Kennedy J., Eberhart R., Particle swarm optimization, in Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ 1995;1942–1948.