*Mathematical and Computational Applications*, Vol. 14, No. 3, pp. 219-228, 2009.


# GENETIC ALGORITHM BASED SOLUTION FOR TSP ON A SPHERE

Aybars Uğur, Serdar Korukoğlu, Ali Çalışkan, Muhammed Cinsdikici, Ali Alp
Department of Computer Engineering, University of Ege,35100
Bornova-Izmir, Turkey
aybars.ugur@ege.edu.tr, serdar.korukoglu@ege.edu.tr, ali@mail.ege.edu.tr
Department of Mathematics, Faculty of Science, University of Ege, 35100
Bornova-Izmir, Turkey
ali.caliskan@ege.edu.tr
International Computer Institute, University of Ege, 35100
Bornova-Izmir, Turkey
muhammed.cinsdikici@ege.edu.tr

**Abstract-** The Traveling Salesman Problem (TSP) is one of the extensively studied combinatorial optimization problems. Various exact or approximation algorithms are devised for solving Euclidean TSP that determine the shortest route through a given set of points in 3-dimensional Euclidean space. In this paper, we proposed a genetic algorithm-based solution for TSP where all points are on the surface of a sphere. A Java-based interactive visualization tool is also developed using Java 3D and optimization results obtained for different problem sizes are presented.

**Key Words-** TSP, Genetic Algorithms, Spherical Geometry, Optimization

## 1.INTRODUCTION

The Traveling Salesman Problem (TSP) also known as the traveling salesperson problem is a well known, popular and extensively studied problem in the field of Combinatorial Optimization and attracts computer scientists, mathematicians, and others. Euclidean TSP which is a NP-hard problem is related with determining the shortest tour through a given set of points in d-dimensional Euclidean space. In metric TSP the nodes lie in a metric space (i.e., the distances satisfy the triangle inequality), whereas in Euclidean TSP the nodes lie in $R^2$ (or more generally, in $R^d$ for some d) and distance is defined using the ℓ2 norm [1].

Many methods based on exact algorithms (branch-and-bound algorithms, progressive improvement algorithms) and approximation algorithms (genetic algorithms, simulated annealing, tabu search, neural nets, ant system, etc.) are devised to find solution for the TSP since 1950s [3] [4] [5]. Many local search methods for finding approximate solutions have been surveyed by [6]. GRASP [7] is iterative two phase search consists of a construction phase and a local search procedure [8]. Some researchers have been performed studies on hybrid evolutionary algorithms for better TSP results which can be found in [9] [10] and [11]. [12] solved 3D-TSP for the multi-dimensional city location. An exact solution for 15,112 German cities from TSPLIB was found in 2001 using the cutting-plane method via linear programming. In March 2005, the TSP of visiting all 33,810 points in a circuit board was solved using Concorde which is a computer code for the symmetric TSP [2].

Genetic algorithm (GA) as a computational intelligence method is a search technique used in computer science to find approximate solutions to combinatorial optimization problems. It applies biologically inspired concepts such as crossover and mutation. Researchers have been tackled the TSP with GAs since 1985.

The k-opt method is one of the most well-known TSP solving local search algorithms. It improves the tour edge by edge and reverses the order of subtour [13]. The complexity of the k-opt algorithm is $O(n^k)$ where n is the number of nodes in TSP. 2-opt procedure was introduced by [14] for the TSP and deletes two edges from a tour and reconnects them those paths in the other possible way (as exchange heuristic), keeping the change if it leads to a better tour.

A sphere is a set of points in three dimensional space equidistant from a point called the center of the sphere (Figure 1). The distance from the center to the points on the sphere is called the *radius* r of the sphere. All the points satisfying the following lie on a sphere of radius r centered at the origin as a function of 3 space coordinates (x,y,z):

$$x^2 + y^2 + z^2 = r^2 \quad (1)$$

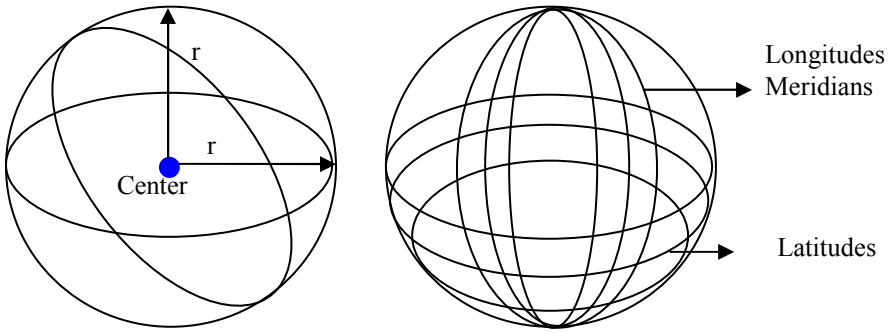


**Figure 1.** Spherical Surface, lines of longitude and latitude.

A great circle is the intersection a plane and a sphere where the plane also passes through the center of the sphere. A geographic example of a great circle is the Earth's equator. Great circles become more important when we realize that the shortest distance between two points on the sphere is along the segment of the great circle. This shortest path is called a geodesic. The curves that minimize the distance between points are called geodesics on any surface. Every meridian of longitude is exactly half a great circle. The parallels of latitude are smaller circles except for the equator.

A path planning study and solution to the TSP on a cuboid can be found in [15]. We proposed a simple and fast algorithm for finding a solution to the TSP on a sphere as a different 3D shape using a hybrid method employing genetic algorithms and 2-opt in this study. This problem differs from the TSP in that all the cities (points) are on a sphere and also it is only allowed to travel on the surface of the sphere. We developed also 3D visualization software for spherical environments.

Some of real-world objects are not ideal shapes. For example, a ball is spherical in shape; it is not ideal or perfect sphere. Approximated shapes and objects similar to

sphere can be found easily around us. The surface of the earth we live on is a good approximation to a sphere. The Earth is not an exact ellipsoid. It is oblate spheroid. Literally, geoid means Earth-shaped. The geoid is a representation and empirical approximation of the surface of the earth. The geoid of Earth can be expressed in the shape of a sphere of revolution. Other planetary bodies including planets, satellites, asteroids and comet nuclei are also good approximations to sphere. Some fruits (orange, apple, melon, pumpkin, etc.), vegetables (eggplant, onion, lettuce, etc.) and seeds are in this form. Geometry of egg has many features in common with that of a sphere. Some furniture parts human-made food products, glass products, plastic products are also spherical.
Sphere is the thermodynamically most stable shape with the lowest overall surface energy. Sphere allows low friction in motion by rolling. Another property of the sphere preferred as wheels can rotate around any axis. A dome is the hollow upper half of a sphere and a common architectural structure element. Hemispherical dome roofs are preferred because of having great deal of structural strength. The sphere has the smallest surface area among all shapes enclosing a given volume. Surface tension minimizes the surface area. So, sphere instances like small water drops and bubbles appear in nature. An isometry of the sphere is a mapping of the sphere to itself which preserves the distance between points. Another example of an isometry is the antipodal map, which maps a point onto the point on the other side of the sphere.

Structures like atoms, molecules, proteins in many research areas like chemistry, biology and physics are represented as spheres. So, solution of spherical TSP can find direct applications for micro and macro systems such as route planning, robot path planning studies based on part picking, part placing.

## 2. GENETIC ALGORITHM BASED SOLUTION FOR TSP ON A SPHERE

TSP on the surface of a sphere is a special case of the standard TSP. Salesperson (or walking insect, robot etc.) can only travel on the surface of the sphere where all points (cities in TSP terminology) lie. Constraint is that points don't lie within the and outside the sphere.

We define the problem as salesperson must travel and a visit N point having coordinates on the surface of the sphere, returning to the starting point, and is required to minimize the total cost of the trip.

A simple and fast method was proposed for finding a solution to the TSP on the sphere:

- Find geodesics (shortest distances) between all pairs of cities on the surface of the sphere.
- Solve TSP problem using genetic algorithms and 2-opt.

Details of the algorithm and method are given in the next subsections.

### 2.1. Representation of a point on the surface of a sphere

The Cartesian description for positions along the path of a curve can be given in parametric form using the following vector point function [16]:

$$P(u) = (x(u), y(u), z(u)) \tag{2}$$

where each of the coordinates is a function of a parameter $u$. In most cases, we can normalize the three coordinate functions so that parameter $u$ varies 0 to 1.0 [16].

We can represent coordinate positions on a surface using the following Cartesian vector point function [16]:

$$P(u,v) = (x(u,v), y(u,v), z(u,v)) \tag{3}$$

Each of the Cartesian coordinates is now a function of the two surface parameters $u$ and $v$. A spherical surface with radius $r$ and center at the coordinate origin can be described with the equations [16]:

$$\begin{aligned} x(u,v) &= r\cos(2\pi u)\sin(\pi v) \\ y(u,v) &= r\sin(2\pi u)\sin(\pi v) \\ z(u,v) &= r\cos(\pi v) \end{aligned} \tag{4}$$

Parameter $u$ describes lines of constant longitude over the surface, while parameter $v$ describes lines of constant latitude [16]. Coordinate positions for different values of parameters $u$, $v$ on spherical surface is shown in Figure 2.

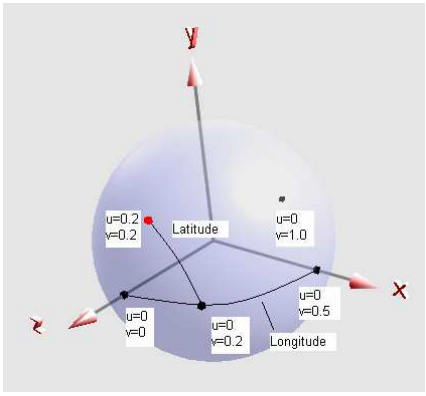


**Figure 2.** Coordinate positions for different values of parameters $u$, $v$ on spherical surface.

We used a unit sphere which is simply a sphere of radius one for simplification of calculations. Results or path lengths can be evaluated easily for spheres of radius $r$, multiplying by $r$.

### 2.2. Finding geodesics between all pairs of points on the surface of unit sphere

Shortest distance between two points $(p_1, p_2)$ on a spherical surface is along the arc of a great circle (Figure 3). So, it can be used the value of angle *theta* $(\theta)$ in radians between two vectors $\vec{v}_1$ and $\vec{v}_2$. Scalar product of two vectors is:

$$\vec{v}_1 \bullet \vec{v}_2 = \left|\vec{v}_1\right|\left|\vec{v}_2\right|\cos\theta \tag{5}$$

where $\theta$ is the angle (smaller one) between two vector directions.
Scalar product is calculated as

$$\vec{v}_1 \bullet \vec{v}_2 = p_{1x}p_{2x} + p_{1y}p_{2y} + p_{1z}p_{2z} \tag{6}$$

Magnitudes of vectors $\vec{v_1}$ and $\vec{v_2}$ are 1 for points on the surface of unit sphere. So shortest distance formula is:

$$\theta = \arccos(\vec{v_1} \bullet \vec{v_2}) \quad (7)$$

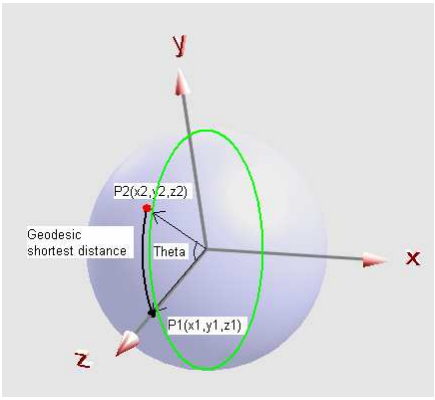


**Figure 3.** *Geodesic*: shortest distance between two points on a spherical surface.

Problem differs from Euclidean TSP, because shortest distance between two points ($p_i$, $p_j$) is calculated by using 3D Euclidean distance (which is a straight line) instead of arc length in 3D Euclidean TSP:

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} \quad (8)$$

Distance from any point $p_i$ to any point $p_j$ is the same as the distance from point $p_j$ to point $p_i$ on the sphere. In this step, $N$x$N$ symmetric distance matrix $D = [d(p_i; p_j)]$ which gives the geodesics between all pairs of points on the spherical surface was computed.

### 2.3. Solving TSP on the Surface of Unit Sphere Using Genetic Algorithms

After obtaining distance matrix, problem is converted to classical TSP. So any optimization method can be used to solve the problem using this matrix. We used a hybrid method that uses GA and 2opt.

**Genetic Algorithm**

*Generate random population of n chromosomes*
*Repeat*
  *Evaluate the fitness f(x) for each chromosome using distance matrix.*
  *Select two parent chromosomes from the population according to their fitnesses.*
  *Generate new population using offsprings (children) through*
    *crossover with* $p_c$ (crossover probability)
    *2-opt mutation with* $p_m$ (mutation probability)
*Until terminating condition*

## 3. JAVA 3D BASED INTERACTIVE VISUALIZATION TOOL

In this study we developed a 3D tool (SphereTSP) to solve this problem using Java and made accessible at http://yzgrafik.ege.edu.tr/~ugur/SphereTSP/. The Java 3D

API (from https://java3d.dev.java.net/) must be installed on the machine and browser before use. Java is an object oriented programming language that takes advantage of the strengths of the Internet [17]. The Java 3D API (a higher level 3D graphics API) which is a standard extension to the Java 2 JDK is an application programming interface used for writing three-dimensional graphics applications and applets. It gives developers high-level constructs for creating and manipulating 3D geometry and for constructing the structures used in rendering that geometry. Application developers can describe very large virtual worlds using these constructs, which provide Java 3D with enough information to render these worlds efficiently.

Java 3D scene-graph is a collection of nodes in a tree-like graph structure. Scene graphs are ideal for 3D graphics applications and games. The scene-graph is an object-oriented data structure which consists of objects to define the geometry, sound, lights, transformations and appearance of visual objects. This structure is constructed of Node objects in parent-child relationships. Group objects allow constructing related scene parts by grouping together one or more child nodes. In our web-based tool, axes, points (cities), path lines, lines of longitude and latitude are some branch groups.

Unit sphere and x, y, z axes are always in the Scene. All three principal axes (coordinate system) are rotated by dragging the left-mouse button over the sphere. Coordinate system is translated (XY plane) via a mouse drag motion by the right mouse button and translated in z direction by Alt-Gr + dragging the left mouse button (zoom effect).

If "Lines" checkbox contains a checkmark, lines of longitude and latitude are drawn on the surface of the sphere (Figure 4a). If unchecked, these lines are removed. "Transparency On" shows all points and path on the sphere (Figure 4b). Only front points which are on the surface from viewing position are visible in "Transparency Off" mode.

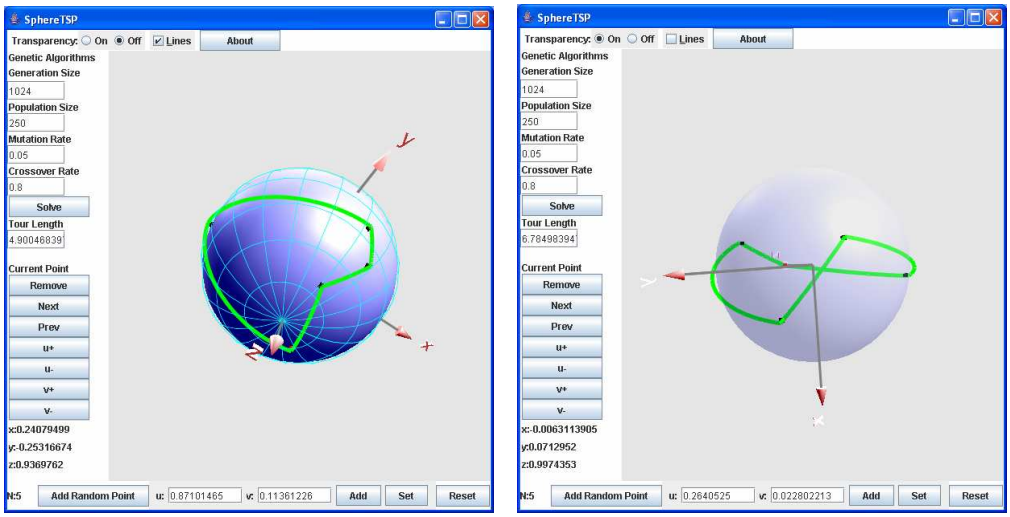


**Figure 4.** Shortest tour on the sphere for 5 points. Transparency is a) off b) on.

GA Parameters can be input by the user. Solution is found quickly for small values of "Generation Size" and "Population Size" parameters. Better results can be obtained for

bigger values of these parameters. “Solve” button solves TSP on the surface of unit sphere using our genetic algorithm based method. Shortest path found is shown on the surface and tour length is set to related Text Field.

“Add Random Point” button inserts a new randomly placed point into the system (on the surface of unit sphere). Some point operations (u+, u-, v+, v-) can be performed on current point. Value of u or v is increased or decreased by 0.02. User can also specify u, v parameter values and add a point to this position using “Add” button, or change the position of the current point on the surface using “Set” button. x, y, z labels provide x, y, z values of current point at any time. User can change the current point using “Next” and “Prev” buttons. Number of points in the system appears at the bottom-left. “Remove” button removes the current point. “Reset” button removes all the points from the system.

## 4. EXPERIMENTAL RESULTS

Simulation results were obtained for N = 100, 150, 200, 250, 300, 350, 400 points on the unit sphere. Simulations were repeated 100 times for each value of N. A new random point set was generated for each trial. This approach is preferred instead of using predefined set of points because of generalizing the results on the unit sphere.

Results which represent optimum tour length were obtained for five different genetic algorithm generation sizes (10, 20, 30, 40, 50 generations). Population size=100, crossover rate=0.80 and mutation rate=0.05 are fixed constant for all experiments. One point crossover was used and 2opt mutation applied. Calculated average values are shown in Table 2. Values are obtained for paths on the surface of a unit sphere.

**Table 2.** Average Spherical TSP tour lengths for N=100, 150, 200, 250, 300, 350, 400 points on the surface of a sphere.

| Generation Size | Number of Points | | | | | | |
|---|---|---|---|---|---|---|---|
| | **100** | **150** | **200** | **250** | **300** | **350** | **400** |
| **10** | 90,1194 | 157,6443 | 227,2337 | 299,1799 | 374,1841 | 441,3127 | 532,6288 |
| **20** | 72,0467 | 132,9872 | 187,7657 | 265,2366 | 333,5504 | 393,6967 | 472,2036 |
| **30** | 53,0306 | 100,9841 | 162,8873 | 224,2341 | 288,0024 | 353,1749 | 439,4960 |
| **40** | 42,6922 | 82,2588 | 141,7211 | 200,0556 | 259,6622 | 323,2888 | 393,7457 |
| **50** | 37,1942 | 70,5737 | 115,1155 | 165,5674 | 226,1183 | 291,1789 | 354,3750 |

Figure 5 shows the average tour lengths for different number of points on unit sphere in the table as a line chart.

If two points are to be visited and these are antipodal points on unit sphere, tour length of Euclidean TSP is 4*r = 4 (attention: returning to the starting point) and tour length of Spherical TSP is 2 * pi = 6,283185 approximately (Figure 6).

Antipodal points cause the maximum difference case between Euclidean and Spherical distances. There are an infinite number of great circles that pass through them for antipodal points. Two points on a sphere that are not antipodal define a unique great circle which traces the shortest path between them.

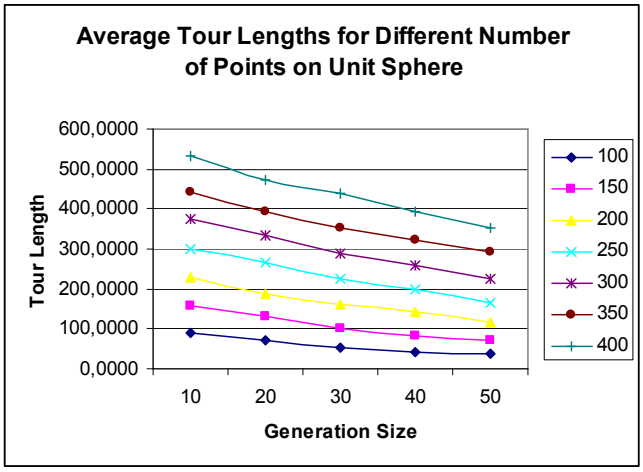


**Figure 5.** Average Tour Lengths for Different Number of Points on Unit Sphere.

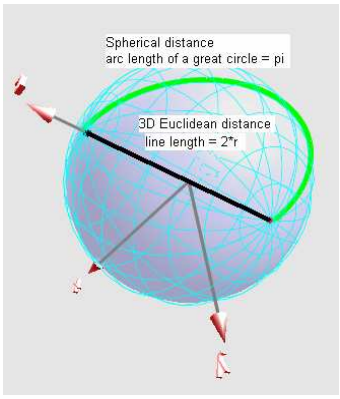


**Figure 6.** Antipodal points, Maximum distance on unit sphere for Spherical TSP and 3D Euclidean TSP

Optimum route found by SphereTSP for 50 points is shown in Figure 7. All points and path are seen at the same time in transparent mode on the left. Solid mode is useful to examine the surface by rotating the sphere system on the right. As in the figure, route found by using optimization techniques may not be the best, especially in large problem sizes.

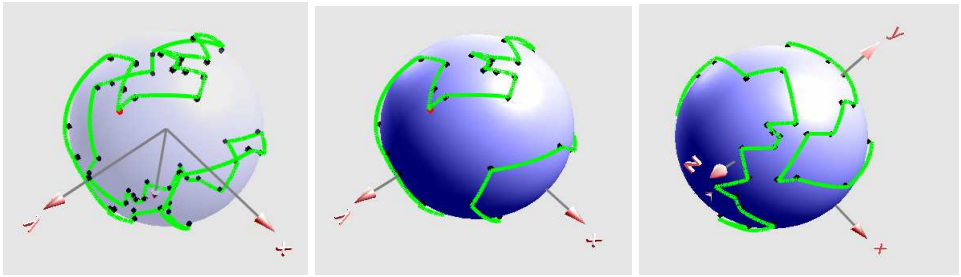

**Figure 7.** Minimum tour obtained on the sphere for randomly placed 50 points

Better results can also be obtained by spending more time for unit sphere by changing values of parameters Generation size, Population size, Crossover rate and Mutation rate.

Increasing generation and population sizes will improve results for more points. Also, genetic algorithm based part of our method can be replaced by any other methods developed or will be developed for solving TSP by considering speed or large problem size issues.

## 5. CONCLUSION

Two important contributions of this paper are to be the first serious application of TSP on sphere which is one of the most common natural object shapes and to propose a simple and efficient genetic algorithm based solving method for spherical TSP which is different from 3-dimensional Euclidean TSP. Planar geometry is sometimes called flat or Euclidean geometry. The geometry of the two dimensional surface of a sphere is spherical geometry which is an example of a non-Euclidean geometry. Lines in plane geometry are replaced by great circles in spherical geometry. For example, minimum distance between points which is a straight line in plane geometry is replaced by geodesics (great circles). Another contribution is to develop java and web based 3D tool to experiment TSP on the surface of the sphere. Future work is to implement and adapt current best exact solution methods (i.e. Concorde) besides genetic algorithms to our method for solving spherical TSP.

Adapting TSP to sphere and method we proposed are important for path planning studies on the surface of planets especially Earth and Mars or satellites especially moon and small ones. Part placing, picking or visiting vehicles (rovers, robots, etc.) determining the shortest tour will be more important in the field of planetary exploration. Autonomous and optimal path-planning by considering energy consumption is important for planetary rovers analyzing or picking rock and soil samples from different parts of these kinds of structures. Method can also be applied to long-range flights on Earth for air cargo services. Path planning on hemispherical crater cavity surfaces is another interesting area for this. Method (and tool) is also useful for understanding and comparing with insect behaviors on spherical structures such as orange and dome roof.

High-level and web-based 3D graphics API's will play important role to develop new 3D optimization and artificial intelligence algorithms. Combining 3D optimization techniques with 3D visualization and computer graphics makes the area more enjoyable and understandable.

## REFERENCES

1. S. Arora, Polynomial time approximation schemes for Euclidean TSP and other geometric problems, *Proc. 37th Ann. Symp. Foundations of Computer Sci.*, IEEE Computer Soc., 2–11, 1996.
2. Wikipedia, The Free Encyclopedia, *http://en.wikipedia.org/wiki/Main_Page*, 2006.

3. L.M. Gambardella and M. Dorigo, Solving Symmetric and Asymmetric TSPs by Ant Colonies, *International Conference on Evolutionary Computation*, 622-627, 1996.
4. K. Shinozawa, T. Uchiyama and K. Shimohara, An approach for solving dynamic TSPs using neural networks, *Neural Networks, IEEE International Joint Conference* **3**, 2450-2454, 1991.
5. Y. Tsujimura, M. Gen, Entropy-based genetic algorithm for solving TSP, *Knowledge-Based Intelligent Electronic Systems* **2**, 285-290, 1998.
6. D.S. Johnson and L.A. McGeoch, The traveling salesman problem: A case study in local optimization, In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, John Wiley & Sons, New York, 215–310, 1997.
7. T. A. Feo and M. G. C. Resende, Greedy randomized adaptive search procedure, *Journal of Global Optimization* **6**, 109–133, 1995.
8. Y. Marinakis, A. Migdalas, P. M. Pardalos, Expanding neighborhood GRASP for the traveling salesman problem, *Computational Optimization and Applications* **32 (3),** 231-257, 2005.
9. Z. J. Lee, A hybrid algorithm applied to travelling salesman problem, *Networking, Sensing and Control, IEEE International Conference* **1**, 237-242, 2004.
10. C. M. White and G. G. Yen, A hybrid evolutionary algorithm for traveling salesman problem, *Congress on Evolutionary Computation (CEC2004)* **2**, 1473-1478, 2004.
11. Y. Marinakis, A. Migdalas and P. M. Pardalos, A Hybrid Genetic-GRASP Algorithm Using Lagrangean Relaxation for the Traveling Salesman Problem, *J. Comb. Optim.* **10(4)**, 311-326, 2005.
12. S. Takahashi, K. Fujimura and H. Tokutaka, The SOM-TSP method for the three-dimension city location problem, *Neural Information Processing* **5**, 2552-2555, 2002.
13. H. Sengoku and I. Yoshigara, A Fast TSP Solver Using GA on Java, *Third International Symposium on Artificial Life, and Robotics (AROB III'98)*, 1998.
14. S. Lin, Computer solutions of the traveling salesman problem, *Bell Systems Journal* **44**, 2245–2269, 1965.
15. A. Uğur, Path Planning On A Cuboid Using Genetic Algorithms, *Information Sciences* **178**, 3275-3287, 2008.
16. D. Hearn and M.P. Baker, *Computer Graphics with OpenGL*, *3rd Edition*, Prentice Hall, 2004.
17. Z. Pan, J. Zhu, W. Hu, H. P. Lun and X. Zhou, Interactive learning of CG in networked virtual environments, *Computers & Graphics* **29**, 273–281, 2005.