Application of Ant Colony Optimization for the Solution of 3 Dimensional Cuboid Structures

Hüseyin Eldem¹, Erkan Ülker²

¹Computer Technologies Department, Karamanoğlu Mehmetbey University, Karaman, Turkey
²Computer Engineering Department, Selçuk University, Konya, Turkey
Email: heldem@kmu.edu.tr, eulker@selcuk.edu.tr

Received November 2013

Abstract
Traveling Salesman Problem (TSP) is one of the most widely studied real world problems of finding the shortest (minimum cost) possible route that visits each node in a given set of nodes (cities) once and then returns to origin city. The optimization of cuboid areas has potential samples that can be adapted to real world. Cuboid surfaces of buildings, rooms, furniture etc. can be given as examples. Many optimization algorithms have been used in solution of optimization problems at present. Among them, meta-heuristic algorithms come first. In this study, ant colony optimization, one of meta-heuristic methods, is applied to solve Euclidian TSP consisting of nine different sized sets of nodes randomly placed on a cuboid surface. The performance of this method is shown in tests.

Keywords
Euclidean TSP; Ant Colony Optimization; Metaheuristic; Path-Relinking; Path Planning; Cuboid

1. Introduction
Traveling salesman problem (TSP) is a well-known problem in combinatorial optimization and it has been widely studied in the theoretical computer science and engineering applications. It is a problem 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 can also be stated as a Hamiltonian cycle problem used in data modeling in computer sciences and investigated in the scope of a graph theory. Since 1950s, many researchers (e.g. computer scientists, mathematicians and others) have studied to develop methods that solve TSP finding the shortest route for cities in a given collection. If all costs between two cities are equal for both cities, TSP is named as symmetric, and otherwise, it is called asymmetric TSP.

TSP applications are frequently encountered in transportation systems. A bus route collecting passengers at certain bus stops can be organized as TSP. Developing effective TSP solutions is gradually gaining importance in route planning problems such as airways, delivery trucks, mail carriers and computer networks.

TSP, which is evaluated in the area of discrete and combinatorial problems, has been widely studied in the area of similar popular problems. In the symmetric TSP, the distance between cities \(i\) and \(j\) is the same, i.e., \(d_{ij} = d_{ji}\). In the asymmetric TSP, this rule may not be valid all the time. Euclidean TSP is special cases of symmetric TSP where the cities are positioned in \(\mathbb{R}^n\) space for some \(m\) values for which the cost function satisfies the triangle equation. The \(\ell_m\) norm, Euclidean norm if \(m = 2\), is defined as \(\sqrt{\sum_{i=1}^{n} (x_i - y_i)^m}\) for \(m \geq 1\) and \((x_1, \ldots, x_m), (y_1, \ldots, y_m) \in \mathbb{R}^m\). Two-dimensional Euclidean TSP is a popular and widely studied issue [1]. There are a great number of publications in literature devoted to TSP solution. Various heuristic algorithms, which give the possible closest solutions to the best ones in a certain time, based on simulated annealing [2], genetic algorithms (GA) [3-5], tabu search [6,7], artificial neural networks [8-10] and ant colony systems [11-16] have been developed. Meanwhile, local search algorithms like 2-opt and 3-opt were also used to solve TSP [17]. In order to attain better results, some researchers tried hybrid evolutionary algorithms [18-21]. Lately, 2-dimensional TSP problem has been transferred to the 3rd dimension and studied in literature. In [22-24], TSP applications were performed for the points on 3D geometric shapes like sphere and cuboid. New algorithms were developed through using genetic algorithms for TSP solution on cuboid geometric shape [22], Particle Swarm Optimization (PSO) on cuboid shape [23] and genetic algorithms on 3D sphere shape [24].

Ant colony optimization (ACO), one of metaheuristic algorithms used to solve optimization problems, was proposed as a PhD thesis by Marco Dorigo in 1992 [15]. It is a probabilistic technique that is used to solve computational problems by investigating the probable ways on the graphs. It originates from the behavior of ants that find the shortest distance between their nests and the food source while seeking food and rapidly choose the shorter path. Various TSP applications were successfully solved by means of ACO techniques.

In the proposed study, TSP for points on a cuboid shape is solved by ACO algorithm. As 2-dimensional TSP is a difficult problem and its reasons are given in the previous studies in literature, the fact that 3D TSP is a difficult problem is not proved in this study. To the knowledge of the author, there is no study in literature that solves 3D TSP with ACO. In the existing TSP problems, coordinates or distances of points are known. The problem in this study differs from the existing TSP problems in that all points are located on a cuboid shape and the transition between points can be made through cuboid surface [22]. The developed 2D Euclidian TSP methods could be directly applied to 3D TSP with points in 3D space. However, TSP on a cuboid shape is different from 3D TSP as minimum distances between points on different surfaces cannot be calculated with Euclidian distance. As the travel takes places on the surfaces of cuboid, routes should be formed on the surfaces, as well, and not within the cuboid. The solutions could be used in route planning and collecting and placing pieces during the creation of a cuboid structure. This optimization method might be needed for small robot agents climbing up the wall, route planning, cleaning and painting. In this study, the performance of the developed method using ACO is tested, also its comparison with GA selected from the literature and advantages are presented. Primarily, the definitions of points on cuboid surface and determination of distances between points are explained. Subsequently, the adaptation of the problem to ACO is explained in the further parts of the paper.

2. Description of Cuboid Shapes

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 (Figure 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 Figure 1, cuboid is one of the three-dimensional objects with height, width and length.

Cuboid objects have six faces: back \((S_b)\), bottom \((S_t)\), left \((S_l)\), front \((S_f)\) and right \((S_r)\) (Figure 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).

\(N\) points could be present on any of 6 cuboid faces. \(N = 7\) cities \((c_0, c_1, c_2, c_3, c_4, c_5, c_6)\) could be located on cuboid faces \((S_b, S_t, S_l, S_f, S_r, S_o, \) and \(S_d)\) as in Figure 3.

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 \(d_{ij}\). Then the distances between all pairs of cities can be computed in an \(N \times N\) 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)
Figure 1. A cuboid shape.

Figure 2. A cuboid object.

Table 1. Geometrical data of a cuboid.

<table>
<thead>
<tr>
<th>Faces</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>Vertices</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$ (Back Face)</td>
<td>$0 &lt; x &lt; \text{Length}$</td>
<td>$0 &lt; y &lt; \text{Height}$</td>
<td>$0$</td>
<td>$0,1,3,2$</td>
<td>$0,2,3,1$</td>
</tr>
<tr>
<td>$S_1$ (Bottom Face)</td>
<td>$0 &lt; x &lt; \text{Length}$</td>
<td>$0$</td>
<td>$0 &lt; z &lt; \text{Width}$</td>
<td>$0,4,5,1$</td>
<td>$0,8,4,9$</td>
</tr>
<tr>
<td>$S_2$ (Left Face)</td>
<td>$0$</td>
<td>$0 &lt; y &lt; \text{Height}$</td>
<td>$0 &lt; z &lt; \text{Width}$</td>
<td>$0,2,6,4$</td>
<td>$1,10,5,8$</td>
</tr>
<tr>
<td>$S_3$ (Front Face)</td>
<td>$0 &lt; x &lt; \text{Length}$</td>
<td>$0 &lt; y &lt; \text{Height}$</td>
<td>$\text{Width}$</td>
<td>$4,5,7,6$</td>
<td>$4,6,7,5$</td>
</tr>
<tr>
<td>$S_4$ (Top Face)</td>
<td>$0 &lt; x &lt; \text{Length}$</td>
<td>$\text{Height}$</td>
<td>$0 &lt; z &lt; \text{Width}$</td>
<td>$2,3,7,6$</td>
<td>$3,11,7,10$</td>
</tr>
<tr>
<td>$S_5$ (Right Face)</td>
<td>$\text{Length}$</td>
<td>$0 &lt; y &lt; \text{Height}$</td>
<td>$0 &lt; z &lt; \text{Width}$</td>
<td>$1,3,7,5$</td>
<td>$2,11,6,9$</td>
</tr>
</tbody>
</table>

Figure 3. View of seven points on cuboid faces.
two cities on adjacent faces. The pseudocode used for calculating the distance between \( c_i \) and \( c_j \) points is given below:

```c
if (face\( c_i \)) == face\( c_j \))//SAME FACE
{
    \( d_{ij} \) = Euclidean distance between the two points;
}
```

```c
else if (face\( c_i \)) (opposite) face\( c_j \))//OPPOSITE
{
    calculate four alternative path distances \( d_i \) (through remaining faces)
    \( d_{ij} = d_0; \)
    for(int k = 0; k < 4; ++k)
        if\( d_k < d_{ij} \) \( d_{ij} = d_k; \)
}
```

```c
else if (face\( c_i \)) (adjacent) face\( c_j \))//ADJACENT
{
    \( d_0 \) = 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. 1) Front-Back \{\( S_3, S_0 \)}, 2) Left-Right \{\( S_2, S_3 \)} and 3) 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  
IV) Left-Bottom-Right  

In this case, the possible routes \( (d_1, d_2, d_3 \) and \( d_4) \) can be calculated, respectively, as follows:

for alternative (I)
\[
\begin{align*}
    d_y &= y_j - y_i \\
    d_z &= (\text{width} - z_j) + \text{length} + (\text{width} - z_i) \\
    d_1 &= \sqrt{d_y^2 + d_z^2}
\end{align*}
\]

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, 1) Front-Bottom, 2) Front-Right-Bottom, 3) Front-Left-Bottom, and 4) Front-Top-Back-Bottom alternatives must be calculated. However, as the nonneighboring Top and Back faces are present for Front-Bottom transition in 4) 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, as follows, and then, the shortest route is chosen.

for alternative (I)
\[
    d_x = x_j - x_i
\]
\[ d_z = (\text{width} - z_j) + y_i \]

\[ d_z = \sqrt{d_x^2 + d_y^2} \]

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.

3. The Solution of TSP on a Cuboid Object by Using ACO

3D TSP that would be applied to the surface of the cuboid is different from normal 2D TSP [22]. The salesman (ant or robot) 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 ACO method.

After creating a distance matrix consisting of each pair of points, problem solution becomes the same as standard TSP. After this stage, the solution of the problem can be examined by each method to solve TSP, which is depicted in the literature survey of the introduction part. In this paper, solutions were obtained for specific number of set of points by using ACO.

**General structure of the ACO:**
initialize all edges to (small) initial pheromone level τ₀;
place each ant on a randomly chosen city;
for each iteration do:
    do while each ant has not completed its tour:
        for each ant do:
            move ant to next city by the probability function and roulette wheel selection
        end;
    end;
    for each ant with a complete tour do:
        evaporate pheromones;
        apply pheromone update;
        if (ant k’s tour is shorter than the global solution) update global solution to ant k’s tour
    end
end

Based on this general structure, initial values of ACO algorithm are set. Accordingly, initial pheromone loads of each side are updated and stored in pheromone matrix. Using the notations in Title 2, distance matrix consisting of distances between each point and every other point is created. In ACO algorithm, each ant giving solution, i.e., agent ant is located random cities initially. In each iteration, the following Equation (1) is used to select the next city to be visited by ants.

\[
p^{k}_{ij} (t) = \frac{\left[ \tau_{ij} (t)^{\alpha} \cdot [\eta_{ij}]^{\beta} \right]}{\sum \left[ \tau_{ik} (t)^{\alpha} \cdot [\eta_{ik}]^{\beta} \right]} \quad \text{if } j \in \text{allowed}_i.
\]  

where, \( \tau_{ij} (t) \) is the density of the mark left at the edge at a time, t. The value of \( \eta_{ij}, \text{i.e., visibility, is equal to } 1/d_{ij}, \) \( d_{ij} \), is the distance between two points on a cuboid. \( \alpha \) and \( \beta \) are two parameters which control the importance of pheromone relative to the visibility. After ants complete their tours, i.e., after each ant finds its solution, evaporation of pheromone and updating processes are performed. By means of evaporation, it could be possible to forget the wrong solutions and give opportunity to the new tours by preventing the accumulation of pheromone. Meanwhile, the best solution given by ants is the one that yields the best displacement globally and give
the best results when the iterations are complete.

4. Experimental Results

Analyses were performed in a computer of Intel Core 2 Duo P8700 2.53 GHz processor and 4 GB RAM. In coding phase of the method, Matlab R2010a programming language was used and called cuboidTSP software. The simulation results obtained through this software were produced for the points \( N = 10, 50, 100, 250 \) on a \( 1000 \times 1000 \times 1000 \) cuboid.

Simulations were repeated 100-times for each value of \( N \). A new random point set was generated for each trial. In this study, this approach was preferred instead of the method given in [22], i.e., utilization of the predefined set of points to generalize the results on the cuboid. The performance of the suggested method was compared to the results reported by [22].

In [22], Aybars reported results for five different \( GA \) generation sizes (20, 40, 60, 80, 100) and the points of \( N = 10, 50, 100, \) and 250 for \( TSP \) application on cuboid through \( GA \). For each generation, Aybars fixed the size of the population as 250. While the mutations of individuals in a population observed in each generation can be defined as evolution, the total evolution is equal to multiplication of the size of the population by the number of generation. In the literature, for the solution of \( TSP \), it is convenient to take the number of ants to the number of cities. For this reason, in the proposed study, the number of ant was taken equal to the number of points in the \( cuboidTSP \). In order to make a comparison with a \( GA \) publication taken from the literature and to have an equal number of evolutions for \( ACO \) with that depicted in the [22] for each generation, the number of tour is determined by using the following equation:

\[
\text{Number of Tours}_{ACO} = \frac{\text{Generation Size}_{GA} \times \text{Population Size}_{GA}}{\text{Number of Ants}_{ACO}}.
\]

The results representing the optimum length of the tour were obtained for different number of evolutions, i.e., 20, 40, 60, 80 and 100 evolutions. For all experiments, the number of ants were equal to that of points and \( \alpha = 1, \beta = 5 \) and \( \rho = 0.50 \) (the coefficient of evaporation) were kept constant. The calculated average tour distances both for \( GA \) and for the proposed \( ACO \) approach are given in Table 2. Meanwhile, mean calculation durations advised for \( ACO \) in this paper are shown in Table 3. The width, height and length of the cuboid were all taken as 1000. A cuboid of \( 1000 \times 1000 \times 1000 \) size is preferred for the sake of understandability and easy assessment of the results.

When the results of \( GA \) and those of \( ACO \) were compared, see in Table 2, it was observed that the results of \( ACO \) algorithm were much more successful than \( GA \) for \( cuboidTSP \). For instance, when evolution number for 150 points is taken as 100, the tour length obtained with \( GA \) could be shortened by 1/3. This rate is even higher for 200 and 250 points. It can be seen in Figure 4 that tour lengths are within \([5000, 30,000]\) for \( GA \) and \([2000, 10,000]\) for \( ACO \).

Through increasing the numbers of tour and evolution in algorithm, the optimum results can be further improved, as can be seen in Table 2 and Figure 4. \( ACO \) parameters are kept constant in the study. By altering the \( ACO \) parameters, better results can be obtained. Apart from \( ACO \), the results can also be improved through using available hybrid methods for \( TSP \) solution.

Table 2. Calculated average \( cuboidTSP \) tour lengths with \( GA \) [22] and \( ACO \) for \( N = 10, 50, 100, 250 \) points on the surface of the cuboid.

<table>
<thead>
<tr>
<th>Evaluation number</th>
<th>10</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>( GA )</td>
<td>5805</td>
<td>3138</td>
<td>12631</td>
</tr>
<tr>
<td>( ACO )</td>
<td>4755</td>
<td>6340</td>
<td>28612</td>
</tr>
<tr>
<td>( GA )</td>
<td>17938</td>
<td>6135</td>
<td>28430</td>
</tr>
<tr>
<td>( ACO )</td>
<td>17845</td>
<td>6055</td>
<td>28293</td>
</tr>
<tr>
<td>( GA )</td>
<td>28252</td>
<td>6039</td>
<td>8818</td>
</tr>
<tr>
<td>( ACO )</td>
<td>9588</td>
<td>9138</td>
<td>8906</td>
</tr>
</tbody>
</table>
Table 3. Calculated average cuboid TSP tour computation times with ACO for $N = 10, 50, 100, 250$ points on the surface of the cuboid.

<table>
<thead>
<tr>
<th>Evolution Number</th>
<th>10</th>
<th>50</th>
<th>100</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2,652,881</td>
<td>1,606,229</td>
<td>3,847,899</td>
<td>1,469,927</td>
</tr>
<tr>
<td>40</td>
<td>5,106,337</td>
<td>315,917</td>
<td>7,626,159</td>
<td>2,891,988</td>
</tr>
<tr>
<td>60</td>
<td>7,788,457</td>
<td>4,778,198</td>
<td>1,156,268</td>
<td>4,416,729</td>
</tr>
<tr>
<td>80</td>
<td>1,052,052</td>
<td>6,570,665</td>
<td>1,587,084</td>
<td>5,872,519</td>
</tr>
<tr>
<td>100</td>
<td>1,388,971</td>
<td>867,242</td>
<td>2,096,042</td>
<td>7,254,003</td>
</tr>
</tbody>
</table>

Figure 4. Average tour lengths for different amount points on the surface of the cuboid founded by GA [22] and ACO solutions.

5. Conclusion

The main contribution of this paper is to show the applicability of TSP solutions cuboid objects and structures of
any size. The aim of the study is to develop a simple yet effective method through testing ACO method frequently used in TSP problems to determine optimum route by visiting all points on cuboid. For this purpose, TSP optimization results for different-sized point sets were obtained in the first place through ACO method for points on a cuboid object.

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 successful 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. And it is observed that 3D techniques are gradually started to be developed. ACO, which can be effectively used in the solution of current 2D TSP by giving optimum results, was also successful for cuboidTSP. When the results of TSP application on cuboid proposed by Aybars in [22] using GA and the results of the method proposed in the current study are compared, it is seen that ACO provides more successful results in cuboid TSP than GA.

In the future studies, 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


