Constructing the Shortest Hamiltonian Circuits and Cycles on a Set of Line Undirected Segments

Natalie D. Ganelina
Faculty of Automatics and Computer Engineering,
Novosibirsk State Technical University,
Novosibirsk, Russia
natalie_ganelina@ngs.ru

Abstract

The present paper is devoted to the problem of constructing Hamiltonian cycles on a set of line segments.

Given a set of undirected disjoint line segments we build a minimal closed tour including all segments only once. The problem is approached by Ant optimization.

Keywords: *Simple circuit, Hamiltonian cycle, Ant colony approach, pheromone*.

1. INTRODUCTION

Given a set of non-intersecting line undirected segments in the plane, we are required to connect the line segments such a way that they form a simple circuit. We call simple circuit 'optimal' (shortest) if it has minimal length among all feasible circuits. In other words, we must choose a route that passes through each segment and the total length of the route is minimal. The problem described is a natural generalization of the problem of finding simple circuits (or Hamiltonian cycles) from a set of points (Traveling Salesman Problem). In general, a set of line segments does not necessarily admit a simple circuit. An example of such a set is given in figure 1.

It was shown that to determine whether a set of segments admits a simple circuit is NP-hard as well as constructing such a circuit in general case.

The task of defining circuits from a set of points is a recurring theme that appears in variety of applications. The benchmark Traveling Salesman Problem can be used in network routing problems, in the area of pattern recognition, etc. Surprisingly, the problem of constructing circuits from a set of line segments has not received much attention.

The task of finding shortest tours on a set of segments can appear, for example, in the collection of sites on a plane, which should be visited in a prescribed order. If in addition, edges in the tour that cross greatly increase the cost function then we need to find a shortest tour with few or no intersections. This is close to the so-called 'tube-passing problem' first stated by Liesegang (1976). Similar problems exist in plotter sequencing systems and metal cutting motion optimization for NC-programs design [1].

Liesegang gives a branch and bound algorithm for the exact solution, but the execution time in this case depends exponentially on the number of segments (tubes). Thus the exact algorithms can not be used in real problems, where the number of lines can be very large and the time is limited.

Later a number of variations of basic problem was analyzed by Frederickson (1978) and all of them were shown to be NP-hard.

Leipala and Nevalainen (1978) suggested a system using a nearest neighbor rule and k-change rule for local perturbation for a plotter sequencing problem [4].

Several papers were aimed to inquire a possibility of constructing circuits from a set of segments. In 1980s David Rappaport [5] investigated segment endpoint visibility graphs and stated that a set of segments admits a simple circuit if each segment has at least one endpoint on the convex hull of the segments.

In 1990s O'Rourke and Rippel [7] proved that there is a Hamiltonian polygon if the supporting line of no line segment crosses any other line segment. Earlier Mirzaian [6] conjectured that every set of disjoint line segments admits a Hamiltonian polygon, that is, there exists a simple closed polygonal paths whose vertices are exactly the endpoints of the line segments and whose sides correspond to edges of the segment endpoint visibility graph. Later this hypothesis was proved.

We are interested in constructing shortest closed cycles (Hamiltonian cycles) on an arbitrary set of line undirected segments by using heuristic methods. Heuristic techniques make it possible to find suboptimal solution with admissible execution time while exact methods in large scale problems perform worse. As a basic model we consider Ant Colony approach [2, 3]. The similar problem approached by genetic algorithms is presented in [1], where segments are formed from insertions on a cutting chart.

The paper is organized as follows. In section 2 preliminary definitions are introduced. The description of Ant colony approach is given in section 3. Some results are presented in section 4. Conclusions and further research lines are given in section 5.

2. PRELIMINARIES

A set of non-intersecting line segments S is represented as $S = (s_1, s_2, \ldots, s_n)$ (to be referred to from now on as segments). The endpoints of S are represented by the set of $2n$ points, $P = (p_1, p_2, \ldots, p_{2n})$. For our algorithm we identify each segment i by its endpoints (p_i, p_{i+1}) or by one point on the segment b_i (from initial point to end point; in the examples presented segments are indicated by middle point if other is not stated).

Given a set of non-intersecting undirected segments represented by S it is sometimes possible to find a simple circuit.

A *simple circuit* is a sequence of edges such that for all edges any two of them can intersect only at their endpoints. In general case segments do not intersect. But we permit segments to intersect at their endpoints (then we have polygonal line).

If such a circuit exists we say that *S admits* a simple circuit. If a circuit is closed we call it a *cycle*.

A simple cycle is *Hamiltonian*. We are to build Hamiltonian cycle with *minimal length*. The length of a cycle includes lengths of segments and distances between segments. The distance between two points is *Euclidean distance*.

A set of segments can be described in terms of graph theory. Segments' endpoints are the *vertices*, a line connecting any two segments (the endpoint of the first segment and the initial point of the second segment) is the *edge*. A set of undirected line segments in terms of our problem is presented by a fully connected undirected planar graph.

3. ANT COLONY APPROACH

The idea of Ant colony optimization is presented in [2, 3]. Here we remind the basic features of the algorithm and describe some parameters that affect the solving process.

Ant optimization is an approach to stochastic combinatorial optimization. The main characteristics of this model are: positive feedback, distributed computation and the use of a constructive greedy heuristic. Positive feedback accounts for rapid discovery of good solutions, distributed computation avoids premature convergence, and the greedy heuristic helps find acceptable solutions in the early stages of the search process.

In the approach discussed the search activities is distributed over so-called '*ants*', that is, agents with very simple basic capabilities which, to some extent, mimic the behavior of real ants. According to the authors of the method [2, 3] the research on the behavior of real ants has greatly inspired their work. One of the problems studied by ethologists was to understand how almost blind animals like ants could manage to establish shortest route paths from their colony to feeding sources and back. It was found that the medium used to communicate information among individuals regarding paths, and used to decide where to go, consists of *pheromone trails*. A moving ant lays some odorous substance – pheromone (in varying quantities) on the ground, thus marking the path by a trail of this substance. While an isolated ant moves essentially at random, an ant encountering a previously laid trail can detect it and decide with high probability to follow it, thus reinforcing the trail with its own pheromone. The collective behavior that emerges is a form of *autocatalytic* behavior where the more the ants following a trail, the more attractive that trail becomes for being followed. The process is thus characterized by a positive feedback loop, where the probability with which an ant chooses a path increases with the number of ants that previously chose the same path. In other words, *Ant colony is a set of cooperating simple agents with collective behavior regulated by positive feedback*. The ants can communicate (cooperate) through pheromone trail.

As we are not interested in simulation of ant colonies, but in the use of artificial ant colonies as an optimization tool, our system will have some major differences with a real (natural) one: artificial ants will have some memory, they will not be completely blind, they will live in an environment where time is discrete. An ant stores in its memory the information of traversed part of a tour. An ant can 'see', that is evaluate how far the point it should move to is and it can evaluate how attractive a path is (calculate how big the amount of pheromone which was laid by preceding ants is).

The idea is that if at a given point an ant has to choose among different paths, those which were heavily chosen by preceding ants (that is, those with a high pheromone trail level) are chosen with higher probability. Furthermore high trail levels are synonymous with short paths.

The method is bi-criterial, it means that solving process is based on two factors: pheromone trail and visibility. A path with high trail and big length (if the following segment is far from the current one) can be more attractive than the close one with small quantity of pheromone and vice versa.

Ants build closed tours by visiting all segments and returning to the initial one. To disable an ant go through a segment twice a special taboo list is used. All ants begin moving at the same time. Each ant evaluates the possibility of choosing the next step by two parameters: how far the next segment is (visibility) and how attractive the path is (pheromone trail). The number of ants is equal to the number of segments on default, ants are distributed randomly. The working memory (taboo list) is emptied at the beginning of each new tour and is updated after each time step by adding the new visited segment.

Pheromone (trail). In our model pheromone trail on the edge indicates that some ants have already passed on this very edge. The more pheromone has been laid on this part of the path the more new ants will choose it (positive feedback effect). We use positive real numbers to simulate pheromone trail which every ant 'lays' on the path between two segments. Calculating the intensity of trail (pheromone), i.e. calculating how attractive this path is, the length of the shortest tour found at the current moment is taken into account, as well as coefficients of pheromone updating. This let the ant manage the probability of choosing the path. Pheromone updating rule in most general case can be presented as (1).

$$\tau_{ij}(t+1) = (1 - g) * \tau_{ij}(t) + g * \Delta\tau_i , \qquad (1)$$

where t is time step, g is an evaporation coefficient (global or local), $\Delta\tau_{ij}$ is the amount of pheromone added (arbitrary constant or a variable proportional to the minimal length of the tour found by the time t). Global updating rule lays a trail only on the edges included in the shortest path. When all the ants have completed a tour the ant that made the shortest tour modifies the edges belonging to its tour. Global trail updating is similar to a reinforcement learning scheme where better solutions get a higher reinforcement.

In local updating we mark edges which are included in a path in the process of constructing a tour at each time step. Local updating is intended to avoid a very strong edge being chosen by all ants. Every time an edge is chosen by ants its amount of pheromone is increased and decreased: we not only add pheromone but also evaporate it, i.e. increase the possibility to find new solutions. Evaporation was also motivated by trail evaporation in real ants.

The amount of pheromone deposited on each visited edge by the best ant is inversely proportional to the length of the tour: the shorter the tour the greater the amount of pheromone deposited on edges. This manner of depositing pheromone is intended to emulate the property of differential pheromone trail accumulation, which in the case of real ants was due to the interplay between length of the path and continuity of time.

Visibility. This is a parameter which corresponds with the distance between segments. We used – as the simplest variant – the value inversely proportional to the distance between two segments considered. If this parameter is dominating solving process the algorithm converges to greedy principle: as the following step the

closest segment is chosen (with all other constraints obeyed). The segment visibility can be presented as (2).

$$\eta_{ij} = f\left(d_{ij}\right), \tag{2}$$

where d_{ij} – the Euclidian distance between two segments i and j.

Each ant in Ant colony algorithm builds a tour which includes all segments it has passed along only once. A segment can be indicated by one point (any point on the segment: 0 – initial point, 1 – end point) or two endpoints. In the first case while constructing tours ants move from any point on initial segment to the corresponding point on the following segment. After the best solution found a tour on initial segments is reconstructed.

In the second model an ant choosing one endpoint of the segment automatically moves to another endpoint of this segment. The number of vertices in such a graph is twice as many as the number of segments; the number of ants is equal to the number of segments.

Identifying a segment by two endpoints shows better results but the first approach is more perspective if we consider constructing Hamiltonian cycles on a set of closed figures (or if we have both figures and line segments).

An artificial ant k at time t in a segment i chooses a segment j to move to among those which do not belong to its taboo list (allowed_k) applying the following probabilistic formula:

$$s = \begin{cases} \max\limits_{j \in allowed_k} \left\{ \left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta} \right\}, & q < q_0 \\ r, & otherwise \end{cases}, \tag{3}$$

where $\tau_{ij}(t)$ is the amount of pheromone trail on the edge between segments i and j at the time t; η_{ij} is the visibility between i and j; α is a sensibility to trail and β is a sensibility to distance (parameters that help regulate the priority of pheromone trail or visibility and proportion between these two factors), q is a random variable [0, 1], q_0 is the knowledge using level (see below); r is a random variable selected according to the following probability distribution, which favors edges which are shorter and have a higher level of pheromone trail:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} * \left[\eta_{ij}\right]^{\beta}}{\sum\limits_{m \in allowed_k} \left[\tau_{im}(t)\right]^{\alpha} * \left[\eta_{im}\right]^{\beta}}, & j \in allowed_k \\ 0, & otherwise \end{cases}, \tag{4}$$

where $p_{ij}^{k}(t)$ is the probability with which ant k chooses to move from i to j.

Knowledge using level q_0. Ant colony optimization is distinguished by other methods of ant optimization by this parameter that can be interpreted as a reinforcement learning system, in which reinforcements modify the strength (i.e. pheromone trail) of connections between segments. An ant can either, with probability q_0, exploit the experience accumulated by the ant colony in the form of pheromone trail or, with probability $(1 – q_0)$, apply a biased exploration (exploration is biased towards short and high trail edges) of new paths by choosing the segment to move to randomly, with a probability distribution that is a function of both the accumulated pheromone trail, the heuristic function, and the working memory, where the information about visited segments is stored.

Adjacent coefficient. We use so called 'adjacent coefficient' to regulate movements between two endpoints identifying a segment. This parameter is close to visibility between segments. Pheromone updating rule is not applied to a segment but to the distance between segments (edges). Increasing an adjacent coefficient we force ants move along a segment (polygonal line, figure). If the requirement of segment pass-through is removed (for example, we can move along a polygonal line, leave it for passing a segment and return), then adjacent force is reduced. In the experiments described below this constraint is not removed.

The cost function in general case is the sum of segments' lengths and distances between the segments from the set of feasible transitions between the segments (see (5)):

$$F(L) = \min\left(\sum_{i=1}^{n} l_i + \varphi \cdot N_{\text{int}} \sum_{i,j=0}^{n} L_{ij} \right), \tag{5}$$

where i is a segment, $i = 0, 1, \ldots, n$,

if index $i = 0$ then it is origin of coordinates, i.e. point (0,0),

l_i is the length of segment i, L_{ij} $(i, j = 0, 1, \ldots, n)$ is the distance between segments or between any two points on the segments i and j, $L_{ij} \in K$, K is a set of feasible movements (tours) $k=(i_1, i_2, \ldots, i_n)$, $(i_1, i_2, \ldots, i_n)$ is an arbitrary permutation of numbers $1, 2, \ldots, n$, N_{int} is the number of intersections in the route found, φ is a penalty coefficient.

Ant colony algorithm employs an exploration strategy with stochastic component (exploration of new paths biased towards short and high trail edges) and encourages new exploration since each path taken has its pheromone value reduced by the local updating rule.

4. EXPERIMENTAL RESULTS

An arbitrary set of segments is presented in figure 2. Solutions achieved by Ant colony approach are presented in figure.3 – 6. The algorithm finds a solution and calculation is stopped after given number of iterations or after user's request. The solution is feasible if a tour doesn't have self-intersections. The solution is optimal or close to optimal if the length of the tour is minimal (the less it is the better solution is) and the number of iterations required to find this solution is acceptable which depends on the problem scale (we estimate the execution time through the number of iterations; if the number of segments is close to the one on the figures presented then admissible number of iterations is less than 100).

The performance of the algorithm strongly depends on the parameters α (sensibility to trail) and β (sensibility to distance). There were tested several values for each parameters, all the other being constant, and the best were chosen.

If $\beta = 0$ the algorithm used only trail information. If $\alpha = 0$ the algorithm behaved as the nearest neighbor heuristic which let us compare the method proposed with greedy heuristic.

The greedy heuristics guide a search process at the early stages of computation and then the algorithm starts exploiting the global information contained in the values τ_{ij} of trail.

Results presented in fig.5 – 6 illustrate the statements mentioned above. Tours in figures 5-6 are not the best, moreover the first one is not admissible since we do not permit self-intersections in the route, but the pictures confirm the efficiency of the Ant colony approach.

The length of the solution *l* and a number of iterations *n* required to achieve it are presented in Table 1.

Some other configurations with solutions found are shown in figures 7 – 8. In both configurations segments were indicated by two endpoints. Hamiltonian cycle on a set of segments with polygonal lines was also successfully found using the technique proposed (see fig.7).

Figure 1: A set of segments without a simple circuit.

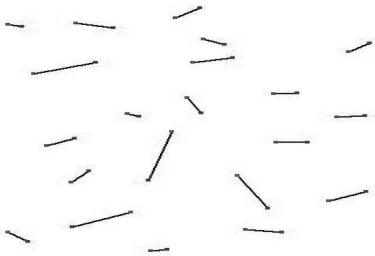

Figure 2: A set of segments.

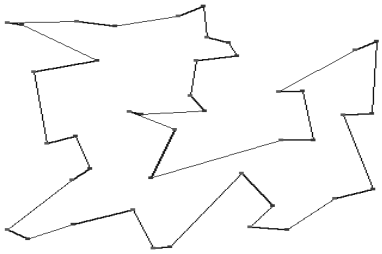

Figure 3: Hamiltonian cycle found. Segments identified by one middle point.

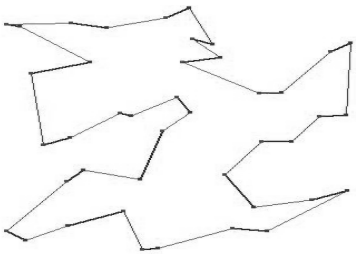

Figure 4: Hamiltonian cycle. Segments identified by two endpoints.

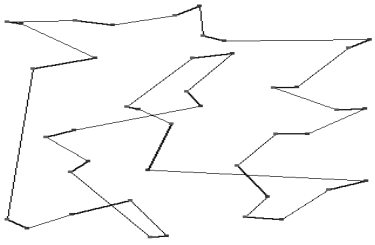

Figure 5: A circuit on a set of segments identified by middle point. Greedy heuristic, β =1, α =0.

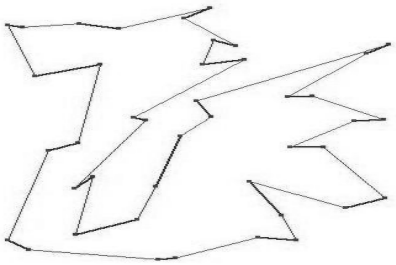

Figure 6: A circuit on a set of segments identified by two endpoints. Greedy heuristic, β =1, α =0.

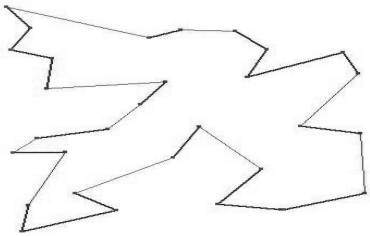

Figure 7: A circuit on a set of segments with polygonal lines.

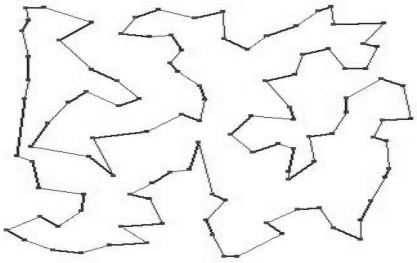

Figure 8: Hamiltonian cycle on a set of segments.

Table 1: Segments identified by two endpoints and middle points, greedy heuristic and standard algorithm

Method	Segments identified by two endpoints		Segments identified by middle points	
	Standard algorithm (fig.4)	Greedy heuristic β =1, α =0 (fig.6)	Standard algorithm (fig.3)	Greedy heuristic β =1, α =0 (fig.5)
Iterations (n)	14	1 900	30	200
Length (l)	2 039	2 397	2 158	2 405

The best solution was found by Ant colony method when segments were identified by two endpoints (2 039 mm). Greedy heuristic took much more computational efforts to find solution close to optimal (see table 1). Identifying segments by one point decreases the number of iterations but leads to the worst solution (a circuit with self-intersections). When the distance between segments is comparable with segments' length or less than it identifying segments by one point unacceptable (solutions found are far from optimal).

It is difficult to compare the results obtained with other ones since the problem was not well investigated. Experiments with segment endpoint visibility graphs (D. Rappaport) or integrally coordinate

segments (J. O'Rourke) reduce greatly a range of segment configurations. Branch and bound method as well greedy principle can be applied to any set of segments but the first one depends exponentially on the number of segments and the second one performs worse as it was shown.

The Ant Colony method was applied to symmetric and asymmetric Traveling Salesman Problems and Quadratic Assignment Problem by its author M. Dorigo [3] and the results were compared with those found by simulated annealing, elastic net, self organizing map and farthest insertion. The technique discussed found solutions which were at least as good as, and often better, than those found by other methods. Though it was not competitive with specialized heuristics (for example, Lin-Kernighan), its performance become very interesting when applied to a slightly different problem. Some extended versions of the algorithm were comparable with hybrid genetic algorithm and other GRASP methods.

All these facts let us conclude that applying Ant colony approach to the problem of constructing Hamiltonian cycle on segments is very perspective. When compared with genetic algorithms on a cutting chart our method performs even better. And according to experiments (see table 1) it can find solutions better than those found by greedy heuristic, moreover solutions are close to optimal and can be found in admissible time.

5. CONCLUSION

We consider the chosen approach very promising for our research. This method is also well suited to parallelization. Solutions found by the method have desirable features such as admissible execution time, stability and closeness to optimum.

The model proposed has the following features: positive feedback, distributed computation, constructive greedy heuristic. Good solutions are discovered rapidly due to positive feedback, distributed computation avoids premature convergence. Acceptable solutions can be found in the early stages of the search process even in large scale problems.

Further work aimed at increasing the efficiency of the approach developed will show how fast a solution can be found. The simplest way to decrease execution time is to use inherent parallelism (distributed computation) mapping it on solving process and parallel architectures.

6. REFERENCES

[1] V. Frolovsky, G. Pushkaryova. *Metal cutting motion optimization for NC-programs design, using genetic algorithms. // Proceedings of the 6th International Conference 3IA'2003 in Computer Graphics and Artificial Intelligence, Limoges (France), May 2003, pp.143-152.*

[2] M. Dorigo, Luca M. Gambardella. *Ant – Q: A Reinforcement Learning approach to the traveling salesman problem. // Proceedings of ML – 95, Twelfth International Conference on Machine learning, Morgan Kaifmann, 1995, pp. 252 – 260.*

[3] Marco Dorigo, Vittorio Maniezzo, Alberto Colorni. *The Ant System: Optimization by a colony of cooperating agents. // IEEE Transactions on Systems, Man and Cybernetics – Part B, Vol. 26, No.1, 1996, pp. 1 – 13.*

[4] T. Leipala and O. Nevalainen. *A plotter sequencing system. The Computer Journal, Vol. 4, No. 22, pp.313--316, November 1979.*

[5] D. Rappaport. *Computing simple circuits from a set of line segments is NP-complete, SIAM J. Comput., 18, 6, 1989, pp.1128–1139*.

[6] A. Mirzaian, *Hamiltonian triangulations and circumscribing polygons of disjoint line segments, Comput. Geom. Theory Appl., 2, 1, 1992, pp. 15–30.*

[7] J. O'Rourke, J Rippel. *Two Segment Classes with Hamiltonian Visibility Graphs. Computational Geometry Theory and Applications, 1994, pp. 209-218.*

About the author

Natalie D. Ganelina is a post-graduate student, Automated Control Systems Department, Novosibirsk State Technical University, e-mail: natalie_ganelina@ngs.ru