Optimal Triangulation Using an Evolutionary Algorithm

Malihe Jahani
Dept. of Computer Science
and IT,
Institute for Advanced
Studies in Basic Sciences,
Zanjan, Iran
m_jahani@iasbs.ac.ir

Bahram Sadeghi Bigham
Dept. of Computer Science
and IT,
Institute for Advanced
Studies in Basic Sciences,
Zanjan, Iran
b_sadeghi_b@iasbs.ac.ir

Abbas Askari
Dept. of Mathematical
Sciences, Sharif
University of Technology,
Tehran, Iran
askari_a@sepanir.com

Abstract

A triangulation of a point set S in the plane is a maximal planar straight-line graph with the vertex set S . In the Minimum Weight Triangulation (MWT) problem, we want to draw a triangulation of a given point set that minimizes the sum of the edges length. Recently, Mulzer and Rote have proved that this problem is NP-hard. In this paper, we present an evolutionary algorithm using Ant Colony Optimization to solve this problem.

1. Introduction

A triangulation of a set S with n points in the plane is a maximal set of nonintersecting edges connecting the points in S . A Minimum Weight Triangulation (MWT) of S is a triangulation with minimum total edge length. Usually, a planar point set has many (exponentially) different triangulations. Note that an MWT needs not to be unique. In the late 70, some significant progresses were made in solving MWT problem, such as Greedy Triangulation and an approximation method using Delaunay Triangulation. It is known that many triangulation algorithms cannot solve the MWT problem correctly [1,2]. The MWT problem has been studied extensively in computational geometry and has applications in computer graphics. This problem has been applicable in cartography, spatial data analysis, and finite element methods. Some optimization criteria for optimal triangulation algorithms are known that include maximizing the minimum angle, minimizing the maximum angle, minimizing the maximum edge length, maximizing the minimum height. Polynomial time algorithms have been developed for determining those triangulations [3,4,5,6]. Some other documents discuss MWT and some other try to solve it approximately [7, 8, 9]. The MWT problem is known as NP-hard [10], and it is not solvable in polynomial time [11]. There is a coding for triangulations which takes advantages of Genetic Algorithm and Annealing Selection for the MWT problem [12].

2. Related works

The literature on the MWT problem is vast, but few significant progresses were made in solving MWT problem, such as well-known method named Greedy Triangulation, an approximation method using Delaunay Triangulation, and heuristic approach using Genetic Algorithm.

In the next subsection we briefly introduce the Minimum Weight Triangulation and Ant Colony Optimization (ACO).

2.1. Minimum Weight Triangulation

A triangulation T of point set S is a maximum set of edges with endpoints in S such that none of edges crosses each other. The set of edges in T are said to triangulate S . The weight of an edge refers to the Euclidean length of the edges. The weight of a triangulation is the total length of its edges. A Minimum Weight Triangulation of a point set S is a triangulation with minimum weight. Figure 1 shows an example of such triangulations.

Figure 1. MWT for 60 points.

2.1.1. Previous Works

Finding a MWT of a finite point set is known to be NP-hard, so unsolvable in polynomial time. The MWT problem was first solved by a Greedy Algorithm which always adds the shortest possible edge to the triangulation [17]. Afterwards, Shamos and Hoey suggested using the Delaunay Triangulation as a Minimum Weight Triangulation [18]. Lloyd showed that in general the Delaunay Triangulation and the Greedy Triangulation are not the Minimum Weight Triangulations. In fact, the Delaunay Triangulation is $\Omega(n)$ times longer than the MWT [19] and the Greedy Triangulation approximates the MWT by a factor of $\Theta(n)$ [20].

There are some efficient algorithms that compute the MWT for restricted classes of point sets. For an n -vertex simple polygon, the algorithms in [13, 14] independently gave a dynamic programming algorithm that computes the MWT in $O(n^3)$ time. Anagnostou and Corneil gave an algorithm to compute the MWT of the vertex set of k nested convex polygons in $O(n^{3k+1})$ time [21]. Hoffmann and Okamoto showed how to obtain the MWT of a point set with k inner points in $O(6^k n^5 \log(n))$ time [22].

Some approaches to compute the MWT for point sets have used the method of finding set of edges that are subgraphs or supergraphs of the MWT. Gilbert showed that the shortest edge always belong to MWT [13]. Also, edges which join mutual nearest neighbors are in the MWT [23]. Trivial edges, which those are not intersected by any other edges and the convex hull of the point set, are subgraphs of the MWT, too. The β -skeleton is also a subgraph of the MWT when $\beta \leq \sqrt{2}$. This subgraph usually contains a lot of disconnected component then in practice it cannot help much in computing the MWT [24]. The LMT-skeleton usually generate more useful results than the β -skeleton [25]. It uses that a MWT is locally minimal triangulation to generate a subgraph of the MWT.

On the other hand, the diamond test generates a supergraph of the MWT [26]. This eliminates all edges except a set of $O(n)$ remaining candidate edges.

In Section 2, we describe MWT problem and Ant Colony Optimization. In Section 3, we present an algorithm to solve the MWT using Ant Colony Optimization. Section 4, presents our experimental results. Section 5, presents our conclusions and future works.

2.2. Ant Colony Optimization

Ant algorithms were inspired by the observation of real ant colonies, which are approximate algorithms used to obtain good enough solutions to hard Combinatorial Optimization (CO) problems in a reasonable amount of computation time. CO is an optimization problem in which, given a finite set of solutions and an objective function that assigns a positive cost value to each of the solutions. In the case

of approximate solution for an NP-hard problem, the goal is finding a good solution in a reasonable amount of time.

Ants are social insects, that they live in colonies and whose behavior is directed more to the survival of the colony. When searching for food, ants initially explore the area surrounding their nest in a random manner. An important and interesting behavior of ant colonies is their foraging behavior, in particular, how ants can find shortest paths between food sources and their nest.

While walking from food source to the nest and vice versa, ants deposit on the ground a substance called pheromone, the quantity of pheromone deposited, which may depend on the quantity and quality of the food. Ants can smell pheromone when choosing their way, and they tend to choose, in probability, paths marked by strong pheromone, the pheromone trail allows the ants to find their way back. Also, it can be used by other ants to find the location of the food source. The indirect communication between the ants via the pheromone trail allows them to find shortest paths between their nest and food sources. This functionality of real ant colonies is exploited in artificial ant colonies in order to solve optimization problems when the closed form optimization technique cannot be applied. In the ACO, the task of each artificial ant is the construction of a feasible solution of the problem and a colony of artificial ants cooperates in finding good solutions. Cooperation is a key design component of ACO algorithms.

Some differences between the behavior of the real ants and the behavior of the artificial ants in ACO algorithms are as follow:

- Artificial ants update pheromone trails only after having generated a solution, but real ants deposit pheromone on the ground whenever they move.
- While real ants cannot memorize any information, each artificial ant has a memory to maintain private information.
- Artificial ants live in the discrete environment and in discrete time.

The ant's memory can use to store useful construction information to compute optimal solution. Also, it can manage the feasibility of the solutions.

In ACO algorithms, a finite size colony of artificial ants with the above described characteristics collectively, searches for good solutions to the optimization problem. Each ant builds a solution or a component of it, starting from an initial state, selected according to the problem.

Characters of the ACO algorithms use the probabilistic model to generate the problem solutions under consideration. The probabilistic model is called the pheromone model. The pheromone model consists of a set of model parameters, which are called the pheromone trail parameters. The pheromone trail parameters have values, called pheromone values. At run-time, ACO algorithms try to update the pheromone values in such a way that the probability to generate high quality solutions increases over time.

The pheromone values are updated using previously generated solutions. The update aims to concentrate the search in regions of the search space containing high quality solutions. In particular, the reinforcement of solution components depending on the solution quality is an important ingredient of ACO algorithms. It implicitly assumes that good solutions consist of good solution components.

Furthermore, the ACO algorithms include a pheromone evaporation mechanism similar to real pheromone evaporation, modifies pheromone information over time. Pheromone evaporation allows the ant colony to slowly forget its past history so that it can direct its search towards new directions without being over-constrained by past decisions. Also, this mechanism avoids a rapid drift of all the ants towards the same part of the search space.

In general, the ACO approach tries to solve an optimization problem by iterating the following steps:

- Constructing candidate solutions.
- Updating pheromone values using the candidate solutions.

ACO algorithms have been successfully applied to diverse combinatorial optimization problems such as TSP, Quadratic Assignment Problem (QAP), vehicle routing, telecommunication networks, graph coloring, constraint satisfaction, Hamiltonian graphs, scheduling, and so on [16].

3. Ant Colony Optimization and MWT

Actually, the MWT can be regarded as a combinatorial optimization problem and the Greedy Triangulation and the Delaunay Triangulation approximate MWT by a factor of point size, S , which is not, closed enough to exact solution. Also, the optimal solution of the available heuristic algorithm based on Genetic Algorithm may be not optimal. The Ant Colony Optimization is better than the Genetic Algorithm in solving most of optimization problem such as Traveling Salesman Problem (TSP). In this section we propose an algorithm using this method to solve the MWT problem. The results of our simulations show that profits of our efforts. In the first subsection, we introduce the algorithm variables and in the next one, we present the ACO_MWT Algorithm.

3.1 Encoding Mechanism for Triangulation

Algorithm variables describe as follows:

η_{ij} , means the inverse of the distance.

m , is the number of ants at each iteration.

ρ , is a coefficient that $(1 - \rho)$ represents the percent of pheromone evaporation.

τ_{ij} , represents the amount of pheromone available on edge (i, j) .

$\Delta\tau_{ij}^k$, represents the amount of pheromone that ant k deposits on edge m .

Adding new pheromone by ants and pheromone evaporation are implemented by the following equation:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij} \quad (1)$$

where $\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k$.

We use an ant structure which contains attributes as follows:

1. *Is_Dead*, which is a Boolean variable that shows ant's living status. An ant may not be able to move to other nodes, so we set ant's *Is_Dead* attribute to *TRUE* and we don't trace the ant's behavior anymore. As a default for each ant, we assume that, ant's *Is_Dead* attribute to *FALSE*.
2. *Tabu_Matrix*, which is a lower triangular matrix($n \times n$), with elements 0, 1 or 2.
 - 2.1. *Ant.Tabu_Matrix*(i, j) = 0, means that the ant can move from node i to neighbor node j .
 - 2.2. *Ant.Tabu_Matrix*(i, j) = 1, means that the ant has already traversed this edge.
 - 2.3. *Ant.Tabu_Matrix*(i, j) = 2, means that the ant cannot move from node i to j , because triangulation will be disturbed.
3. *Current_Node*, which is an integer variable with values between 1 to n , represents the ant's location.

Next_Node is a variable that represents next node that ant must move to it. α and β are parameters that control the relative weight of pheromone trails and heuristic values.

3.2 Algorithm

For a given point set in the plane, at the start of the algorithm, we draw a complete straight-line weighted graph using Euclidean distance. For all edges, (i, j) , the pheromone values, τ_{ij} , are all initialized to a constant value $c > 0$. Note that, we cannot start with pheromone values of 0, because this would lead to a division by zero in Equation 2.

The process of constructing solutions can be regarded as a walk on the fully connected graph whose vertices are the point set that we want to compute MWT on it.

At the start of each iteration, all ants are initially placed in start nodes. The task of each ant is the construction of a feasible triangulation of the given point set. Each ant tries to construct a solution as follow: each ant move from *Ant*[k].*Current_Node* to another feasible neighbor node (*Next_Node*). The ant chooses the *Next_Node* with probability $P_{ij}^k(t)$ as in

$$P_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in allowed_k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Where $allowed_k$ is the feasible neighbor of node i for ant k .

In other words, ants try to avoid intersections. So, when no unvisited edge are faced the ant, $Next_Node = 0$, the ant cannot move to another node. In this case, the ant must be died, so we set Is_Dead property for ant to be *TRUE*.

Otherwise, if $Next_Node \neq 0$ we move the ant from the *Current_Node* to the *Next_Node* and we set $Tabu_Matrix(Current_Node, Next_Node) = 1$ to show that, this edge has already been seen in triangulation and this traversed edge is added to the solution under construction. Then we obtain edges which are intersected with edge $(Current_Node, Next_Node)$ and set the values of these edges in *Tabu_Matrix* with 2 (These edges will not be seen in triangulation).

We move the live ants on the edges until they have completed their triangulation. Once, all ants have completed the construction of their solution, pheromone trails on the paths that are traversed with live ants update as Equation 1. Then we kill all the live ants and for starting a new loop make m live ants (See ACO_MWT Algorithm).

3.3. ACO_MWT Algorithm

Input: a point set in the plane.

Output: a good solution for MWT.

- 1) Initialize
 - (a) Set $NC = 0$ 'NC is the cycle counter'
 - (b) For every edge (i, j) set the initial value $\tau_{ij} = c$ for trial intensity and $\Delta\tau_{ij} = 0$.
- 2) For $k = 1$ to m do
 - (a) Place the m ants on the n nodes and set starting node of the k^{th} ant in $Ant[k].Current_Node$.
 - (b) For every edge (i, j) set $Ant[k].Tabu_Matrix[i, j] = 0$.
 - (c) $Ant[k].Is_Dead = FALSE$.
 - (d) Set $s = 1$'s is edges counter'
- 3) Repeat until $s \neq$ (number of edges of MWT for n nodes)
 - (a) Set $s = s + 1$
 - (b) For $k = 1$ to m do
 - (i) If (Not $Ant[k].Is_Dead$) Then
Choose the $Next_Node$ to move, with probability P_{ij}^k given by equation (2) where $i = Ant[k].Current_Node$ and $j = Next_Node$
 - (ii) If ($Next_Node = 0$) Then
 $Is_Dead = TRUE$
Else
For every node (i, j) that intersect with edge $(Ant[k].Current_Node, Next_Node)$
 - Set $Ant[k].Tabu_Matrix[i, j] = 2$
 - Set $Ant[k].Tabu_Matrix[Ant[k].Current_Node, Next_Node] = 1$
 - Set $Ant[k].Current_Node = Next_Node$
- 4) For $k = 1$ to m do
 - (a) If (not $Ant[k].Is_Dead$) Then

- (i) Compute the L_k of the triangulation described by the k -th ant.
- (ii) Update the shortest triangulation found
- (b) For every node (i, j)
- (i) For $k = 1$ to m do
 - If (not $Ant[k].Is_Dead$) then

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{if } Ant[k].Tabu_Matrix[i, j] = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^k$$

- 5) For every node (i, j) compute τ_{ij} according to equation (1).
 - (a) Set $NC = NC + 1$
 - (b) For every node (i, j) set $\Delta\tau_{ij} = 0$.
- 6) If ($NC < NC_{max}$) and (not stagnation behavior) then
 - (a) For every node (i, j) set $Ant[k].Tabu_Matrix[i, j] = 0$
 - (b) Go to step 2
- Else
 - (a) Print shortest triangulation
 - (b) Stop.

4. Experimental Results

As we show in examples, to evaluate our results of ACO_MWT compared them with exact MWT and Greedy Triangulation (GT) algorithm. Figure 2 shows the optimal solution of the proposal algorithm for point set with seven points is equivalent with MWT.

Figure 2. A point set and difference optimal solutions of the ACO_MWT Algorithm (a), exact MWT (b), and the Greedy Triangulation (c).

In Figure 3 the point set is similar to a set of points introduced by Levkopoulos [27]. In this set of points computed triangulation with ACO_MWT is equal with exact MWT and weight of triangulation is 11427.477, but weight of triangulation that generated by GT is 13956.031.

Figure 3. The results of the ACO_MWT (a), MWT (b), and GT (c).

The triangulation of thirty five points that selected randomly is shown in Figure 4. The weight of the triangulation generated by GT is 4792.306 and MWT and weight of the triangulation generated by ACO_MWT are 4713.256.

Figure 4. The results of the ACO_MWT (a), MWT (b), and GT (c).

In nearly all samples our results are equal with exact MWT. As the figures show, there are many differences between triangulations that generated by GT and ACO_MWT. Whereas MWT is a problem with exponential time, our algorithm computes the good solution in a reasonable time using advantages of Ant Colony Optimization.

5. Conclusion and Future Works

The problem of computing an optimum triangulation with the minimum weight was a longstanding open problem in computational geometry. Recently, Mulzer and Rote have proved that this problem is NP-hard [10]. In this paper, we present a evolutionary algorithm using Ant Colony Optimization to solve this problem. One can solve this problem using other methods or use the results of this paper for some applications in computer graphics.

6. References

- [1] E.L. Lloyd, "On triangulations of a set of points in the plane", In *Proc.18th Ann.IEEE Conf. on the Foundations of Computer Science*, 1977.
- [2] C.A. Rogers "Packing and Covering", *Cambridge University Press*, London, 1964.
- [3] D.M. Mount and A. Saalfeld, "Globally-equiangular triangulations of co circular points"
- [4] R. Sibson, "Locally equiangular triangulations", *Computer Journal* 21, 1978, pp. 243-245.
- [5] H. Edelsbrunner, T.S. Tan, and R. Waupotitsch, "A polynomial time algorithm for the minmax angle triangulation", *6th ACM Symp.Computational Geometry*, 1990, pp. 44-52.
- [6] D. Eppstein, "The farthest point Delaunay triangulation minimizes angles", *Report ICS-90-45*, Univ. California, Irvine, 1990.
- [7] W. Lenhart and G. Liotta, "The draw ability problem for Minimum Weight Triangulations", *Theoretical Computer Science* 270, 2002, pp. 281-286.
- [8] D. Eppstein, "Approximating the Minimum Weight Triangulations", *Tech Report 91-55*, Univ. California, 1991.
- [9] M.B.Dillencourt, "Finding Hamiltonian Cycles in Delaunay Triangulations is NP-Complete", 2130, 1994. doi:10.1126/science.1065467, Univ California.
- [10] W. Mulzer and G. Rote, "Minimum-Weight Triangulation is NP-hard", *Journal of the ACM*, Vol. 55, No. 2, Article 11, 2008.
- [11] M.R. Garey and D.S. Johnson, "Computers and intractability: a guide to the theory of NP-completeness", *W.H. Freeman*, 1979.
- [12] Yu-Bei Lin, Ruo-Mei Wang, and Yi Li, "Minimum Weight Triangulation Based on Improved Genetic Algorithm with Annealing Selection", *Journal of Information & Computational Science* 1: 2, 2004, pp. 337-342.
- [13] P.D. Gilbert, "New results in planar triangulations." *Master Thesis*, University of Illinois, Urbana, 1979.
- [14] G. Klincsek, "Minimal triangulations of polygonal domains", *Annals Discrete Math.* 9, 1980, pp. 121-123.
- [15] E. Anagnostou and D. Corneil, "Polynomial-time instances of the Minimum Weight Triangulation problem", *Comput. Geom.* 3, 1993, pp. 247-259.
- [16] M.Dorigo, G.D. Caro, and L.M. Gambardella, "Ant Algorithms for Discrete Optimization", *Artificial Life* Vol.5, No.3, 1999, 137-172.
- [17] R.D. Duppe and H.J. Gottschalk, "Automatische Interpolation von Isolinien bei willkürlich verteilten Stützpunkten", *Allgemeine Vermessungs-Nachrichten* 77, 10, 1970, pp. 423-426.
- [18] M. I. Shamos and D. Hoey, "Closest-point problems", In *16th Annual Symposium on Foundations of Computer Science. IEEE Computer Society*, Long Beach, Calif., 1975, pp. 151-162.
- [19] D. G. Kirkpatrick, "A note on Delaunay and optimal triangulations", *Inf. Process. Lett.* 10, 3, 1980, pp. 127-128.
- [20] G.K. Manacher and A.L. Zobrist, "Neither the greedy nor the Delaunay triangulation of a planar point set approximates the optimal triangulation", *Inf. Process. Lett.* 9, 1, 1979, pp. 31-34.
- [21] E. Anagnostou and D. Corneil, "Polynomial-time instances of the minimum weight triangulation problem", *Computational Geometry Theory Application* 3, 5, 1993, pp. 247-259.
- [22] M. Hoffmann and Y. Okamoto, "The minimum weight triangulation problem with few inner points", *Comput. Geom. Theory App.* 34, 3, 2006, pp. 149-158.
- [23] B.T. Yang, Y.F. Xu, and Z. You, "A chain decomposition algorithm for the proof of a property on minimumweight triangulations", In *ISAAC 94: Proceedings of the 5th Inter. Symp. on Algorithms and Computation. Lecture Notes in Computer Science*, Vol. 834, Springer-Verlag, London, 1994, pp. 423-427.
- [24] J.M. Keil, "Computing a subgraph of the minimum weight triangulation", *Computational Geometry. Theory Appl.* 4, 1, 1994, pp. 13-26.
- [25] M.T. Dickerson, J.M. Keil, and M.H. Montague, "A large subgraph of the minimum weight triangulation", *Discrete Comput. Geom.* 18, 3, 1997, pp. 289-304.
- [26] G. Das and D. Joseph, "Which triangulations approximate the complete graph", In *Proc. of the Inter. Symp. on Optimal Algorithms. Lecture Notes in Computer Science*, vol. 401, Springer-Verlag, New York, 1989, pp. 168-192.
- [27] C. Levcopoulos, "An $\Omega(n)$ lower bound for the nonoptimality of the Greedy triangulation", *Inform. Process. Letter*, Vol. 25, pp. 25-31, 1987.