Computing the Minimum Weight Triangulation of a Set of Linearly Ordered Points

(Extended Abstract)

Henk Meijer and David Rappaport
Department of Computing and Information Science
Queen's University
Kingston, Ontario
K7L 3N6

Introduction

Consider a set of points, P, in the plane. A triangulation P is a partition of the plane by joining the points in P with non-crossing straight line segments so that every region interior to the convex hull of P is a triangle. Although there are many polynomial time algorithms to obtain the triangulation of a set of points, there is no known polynomial time algorithm to obtain a triangulation that minimizes the sum of the edge lengths of the triangulation [Pr&Sh]. Let us denote such a triangulation by MWT, that is, it is a minimum weight triangulation. This seemingly innocuous problem has proved to be one of the most perplexing problems in combinatorial optimization. Not only is there no polynomial time algorithm to obtain an MWT for an arbitrary set of points, the problem is also not known to be NP-hard [Ga&Jo].

Various aspects of this problem have been explored. A possible avenue of investigation is to look at approximation schemes. In this scenario a polynomial time approximation algorithm is proposed with the intent to produce results that are nearly optimal. Letting C(T) denote the sum of the edge lengths of a triangulation T, one can measure how near an approximate solution is to the optimal solution by evaluating the ratio C(AP)/C(MWT), where AP denotes a triangulation obtained from an approximation algorithm. The two most common approximations are the so called Greedy Triangulation (GT) and Delauney Triangulation (DT) [Pr&Sh]. For each of these methods it has been conjectured that C(AP)/C(MST) = 1. Unfortunately there have since been examples that show that $C(DT)/C(MWT) = \Omega(N)$ [Ki] and $C(GT)/C(MWT) = \Omega(n^{1/2})$ [Le], where n denotes the size of the problems. In [Li] it has been shown that on average C(DT)/C(MWT) and C(GT)/C(MWT) are in O(log n).

An alternate direction is to look for exact solutions for restricted classes of input. An $O(n^3)$ time algorithm to obtain an MWT form a set of vertices of a simple polygon was given independently in [Gi] and [Kl]. Another class of input that admits a polynomial time solution was recently proposed by Anagnostou [An]. Let λ denote a set of k lines. Consider a set of n points, P, lying on the lines λ such that

- no two lines in λ intersect in the interior of the convex hull of P
- no line is vertical
- the lines can be numbered from 1 to k such that all points of P on line i are above line i + 1, and below line i - 1.

We call a set of points P with the above property *linearly ordered*. An $O(n^{3k})$ time and $O(n^{2k})$ space algorithm is given in [An] to compute an MWT of a set of points that are linearly ordered. In this paper we describe an $O(n^k)$ time and space algorithm to obtain an MWT of a set of linearly ordered points.

Preliminaries

Consider a set, P, of n linearly ordered points lying on k lines $\lambda = (\lambda_1, \lambda_2, ..., \lambda_k)$. We use n_i to denote the number of points in P that lie on the line λ_i. For each line $\lambda_i \in \lambda$ consider the smallest closed subset, L_i, that contains all n_i points in P that lie on λ_i. Let x_{i1} and x_{in_i} denote the endpoints of L_i, such that x_{i1} is the leftmost point on L_i. For each point $p \in P$ on L_i we say that it is of rank j on L_i if there are j-1 points in P on the line segment $[x_{i1}, p)$. We can use this indexing scheme to identify the points P, that is, each point is uniquely labelled x_{ij}, denoting that the point is of rank j on L_i. An example with k=4 and N = 17 is shown in figure 1.

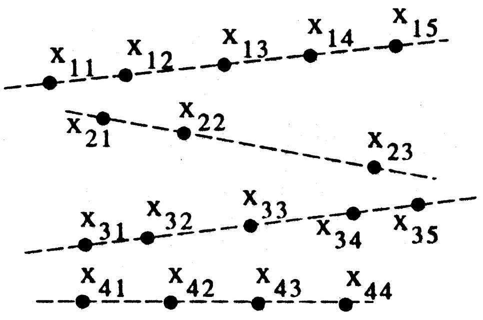

Figure 1.

Definition 1: A contour $Y = (y_1\ y_2 ... y_t)$ is a set of points on the line segments L_i with

$$y_1 = x_{1j} \text{ for some j with } 1 \le j \le n_1$$
$$y_t = x_{kj} \text{ for some j with } 1 \le j \le n_k$$
$$y_s = x_{ij} \text{ and } s < t \rightarrow y_{s+1} = x_{uv} \text{ for some u,v with } u > i.$$

In other words, a contour is a set of at most k points, with at most one point on each segment L_i and including one point each from L_1 and L_k. The number of contours is:

$$n_1.(n_2 + 1).(n_3 + 1)...(n_{k-1} + 1).n_k = O(n^k).$$

Given a contour $Y = (y_1\ y_2 ... y_k)$, let $\Pi(Y)$ denote the chain of line segments obtained by connecting y_i to y_{i+1} for i = 1 to t-1.

Definition 2: A front $Z(Y) = (z_1\ z_2 ... z_k)$ corresponding to a contour Y is a set of points where z_i is the point on the line L_i formed by intersecting $\Pi(Y)$ with λ. We can see that a point z_i in Z(Y) is either a point y_j in Y or is the intersection of L_i and a line segment from y_j to y_{j+1} in $\Pi(Y)$.

Figure 2 shows a contour of size 3 and a front of size k=4.

Since each contour corresponds to exactly one front, it follows from lemma 1 that the number of fronts is $O(n^k)$. We induce a partial ordering on the fronts so that a dynamic programming approach can be used to build up the minimum weight triangulation in stages.

We use $p < z_i$ to denote that p is to the left of z_i and $p \le z_i$ to denote that $p < z_i$ or $p = z_i$.

Definition 3. A front $Z = (z_1\ z_2 ... z_k)$ is said to be lexicographically smaller than the front $W = (w_1\ w_2 ... w_k)$ if for some j, $z_i = w_i$ for $1 \le i < j$ and $z_j < w_j$.

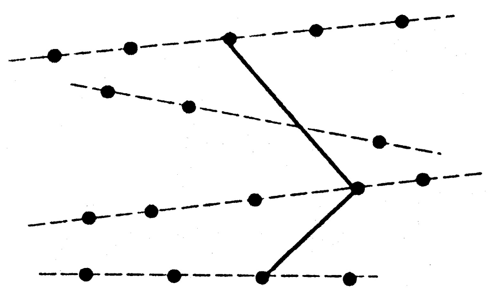

Figure 2.

__Definition__ 4: A contour Y is lexicographically smaller than contour Y' if the front Z(Y) is lexicographically smaller than the front $Z(Y')$.

For a contour Y let T(Y) denote a minimum weight triangulation of the points $\{x_{ij} \mid x_{ij} \leq z_i, z_i \in Z(Y)\}$ and constrained to lie to the left of $\Pi(Y)$.

Algorithm Triangulate

Input:	A set of n points, P, lying on k linearly ordered lines.
Output:	A minimum weight triangulation of P.
Step 1.	Construct a lexicographically ordered list, Q, of contours.
Step 2.	For each contour Y in Q determine the front $Z(Y) = (z_1,...z_k)$ and compute T(Y).

Observe that a triangulation T(Y) for the lexicographically largest contour Y is a triangulation of P.

We begin by showing how to construct Q in $O(n^k)$ steps. Construct an auxiliary set of point X(P) by adding to the set P:

- the midpoints of the line segments (x_{ij}, x_{ij+1})
- a point to the left of x_{i1} for each line L_i $i = 2, k\text{-}1$
- a point to the right of x_{in_i} for each line L_i $i = 2, k\text{-}1$

Thus $|X(P)| = 2n + k\text{-}4$.

Consider a k point contour, C, of X(P). Let $w(C) = (w_1, w_2...w_k)$ denote a k character word such that w_i is the rank of c_i on the line L_i. Using w(C) for each k point contour C we can construct an ordered list, S, in $O(n^k)$ steps. We can construct Q from S in an additional $O(n^k)$ steps. To complete the dynamic programming algorithm, we show that step 2 of algorithm Triangulate can be executed in constant time. In other words, given minimum weight triangulations for all lexicographically smaller contours, we compute the triangulation of a new contour in O(1) time. This is shown by the following principle of optimality condition.

__Lemma__: Every triangulation T(Y) contains as a subset $T(Y')$ where $Z(Y') < Z(Y)$, and there is no other contour Y'' so that $Z(Y') < Z(Y'') < Z(Y)$.

We can now elaborate on step 2 of algorithm Triangulate. For each contour Y we consider all of the (at most 2k-2) configurations of lemma 2. Let D denote an array indexed from 1 to nS, where nS is the size of the ordered list S introduced earlier, so: $nS = (2n_1\text{-}1)(2n_2+1)(2n_3+1)...(2n_{k-1}+1)(2n_k\text{-}1)$.

We store the weight of contour Y in D by using w(C) corresponding to Y as in the explanation of algorithm Triangulate. For each configuration we use a table lookup into Q to find the cost of the minimum weight triangulation of the appropriate lexicographically smaller contour. We then compute the cost of T(Y) by adding the weights of the edges required to complete the triangulation. Therefore, step 2 requires O(1) time per contour, resulting in an $O(n^k)$ minimum weight triangulation algorithm.

We now state the main result of the paper.

Theorem: A minimum weight triangulation of a set of linearly ordered points lying on k lines can be obtained in $O(n^k)$ time and space.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada under grants A-0282 and A-9204.

References

1. [An] Efthymios Anagnostou, "Progress in Minimum Weight Triangulation", M.Sc. Thesis, Department of Computer Science, University of Toronto, available in Technical Report 232/90 (1990).

2. [Ga&Jo] M.R. Garey and D.S. Johnson, Computer and Intractability-A Guide to the Theory of NP-Completeness, W.H. Freeman, (1979).

3. [Gi] P.N. Gilbert, "New results in Planar Triangulations", M.Sc. Thesis, Coordinated Science Laboratory, University of Illinois, Urbana, Illinois, 1979.

4. [Ki] D.G. Kirkpatrick, "A Note on Delauney and Optimal Triangulations", *Information Processing Letters* 10 (1980) 127-128.

5. [Kl] G.T. Klincsek, "Minimal Triangulations of Polygonal Domains", *Ann. Discrete Math* 9 (1980) 121-123.

6. [Le] C. Levcopoulos, "An $\Omega(N^{1/2})$ lower bound for the non-optimality of the greedy triangulation", *Information Processing Letters* 25 (1987) 247-251.

7. [Li] A Lingas, "The greedy and Delauney triangulations are not bad in the average case", *Information Processing Letters* 22 (1986) 25-31.

8. [Pr&Sh] F.P. Preparata and M.I. Shamos, Computational Geometry, An Introduction, Springer Verlag (1985).