New Results for the Minimum Weight Triangulation Problem

By Lenwood S. Heath and Sriram V. Pemmaraju

TR 90-65

NEW RESULTS FOR THE MINIMUM WEIGHT TRIANGULATION PROBLEM

Lenwood S. Heath Sriram V. Pemmaraju

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

December 31, 1990

Abstract

Given a finite set of points in a plane, a triangulation is a maximal set of non-intersecting line segments connecting the points. The weight of a triangulation is the sum of the Euclidean lengths of its line segments. Given a set of points in a plane, the minimum weight triangulation problem is to find a triangulation whose weight is minimum. No polynomial time algorithm is known to solve this problem, and it is also unknown whether the problem is NP-hard. The current best polynomial time approximation algorithm produces a triangulation that can be $O(\log n)$ times the weight of the optimal triangulation. We propose an algorithm that triangulates a set P, of n points in a plane in $O(n^3)$ time and that never does worse than the greedy triangulation. The algorithm produces an optimal triangulation if the points in P are the vertices of a convex polygon. The algorithm has the flavor of a heuristic proposed by Lingas and an analysis similar to his can be performed for our algorithm also, but experimental results indicate that our algorithm performs much better than the heuristic of Lingas. The results of experiments comparing the optimal triangulation with the performance of our algorithm, the heuristic of Lingas, and the greedy algorithm lead us to conjecture that the triangulations produced by our algorithm are within $O(1)$ of an optimal triangulation. We investigate issues of local optimality pertaining to known triangulation algorithms. We define the notion of k-optimality which suggests an interesting new approach to studying triangulation algorithms. We restate the minimum weight triangulation problem as a graph problem and show the NP-hardness of a closely related graph problem. Finally, we show that the constrained problem of computing the minimum weight triangulation, given a set of points in a plane and enough edges to form a triangulation, is NP-hard. These results are an advance towards a proof that the minimum weight triangulation problem is NP-hard.

Key words. Minimum weight triangulation, greedy triangulation, Delaunay triangulation, minimum spanning tree, local optimality, NP-hardness.

1 Introduction

Let $P = \{p_i : i = 1,2,\ldots,n\}$ be a set of points in a plane, where each point p_i has the coordinates (x_i, y_i). To simplify our exposition, we assume that no three points in P are collinear. Let (p_i,p_j) where $i \neq j$ denote the line segment with endpoints p_i and p_j. Let $E(P)$ denote the set of line segments with endpoints in P, given by $E(P) = \{(p_i,p_j) : i \neq j\}$. We often think of the points in P as vertices and the line segments in $E(P)$ as edges of a graph and define various graph problems related to the minimum weight triangulation problem. Two line segments *cross* if they intersect at a point that is not a common endpoint. A *triangulation* $T(P)$ is a maximal set of mutually non-crossing line segments. Let $CH(P)$ denote the set of line segments bounding the convex hull of P, and let $|S|$ denote the cardinality of a set S. Then two properties of any triangulation $T(P)$ are

1. $CH(P) \subseteq T(P) \subseteq E(P)$, and
2. $2n - 3 \leq |T(P)| \leq 3n - 6$.

The *weight* of a line segment (p_i,p_j), denoted by $w(p_i,p_j)$, is the Euclidean distance between p_i and p_j and is given by $w(p_i,p_j) = ((x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$. The *weight* of a triangulation $T(P)$ is given by $W(T(P)) = \sum_{(p_i,p_j)\in T(P)} w(p_i,p_j)$. A *minimum weight triangulation* of a set of a planar set of points P is a triangulation of P that has minimum weight among all triangulations. The minimum weight triangulation problem is

> Given $P = \{p_i : i = 1,2,\ldots,n\}$, a set of n points in a plane, find a minimum weight triangulation of P.

We shall denote an arbitrary minimum weight triangulation of P by MWT(P) and its weight by W(MWT(P)).

To illustrate these concepts, Figure 1 shows a minimum weight triangulation and an arbitrary triangulation of a set of 10 points.

The minimum weight triangulation problem has applications in the numerical approximation of bivariate data. Davis and McCullagh [3] suggest an approach called the *polyhedral approach* to calculate the value of a function f at any arbitrary point p, given the value of f at irregularly spaced points p_i, for $i = 1,2,\ldots,n$. In this approach, the function surface is

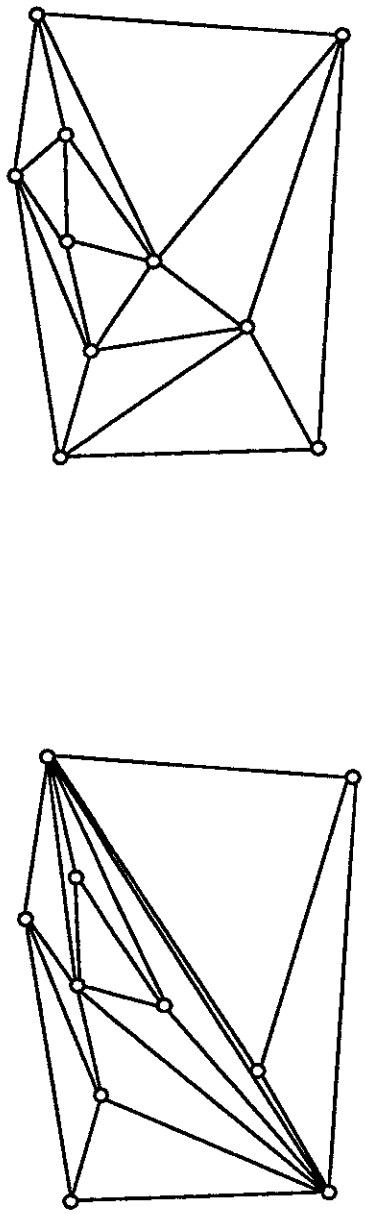


Figure 1: A minimum weight triangulation and an arbitrary triangulation for a set of 10 points.

approximated by a triangulation of the points p_i, for $i = 1, 2, \ldots, n$. The point p lies within some face of the triangulation and $f(p)$ can be approximated by the linear interpolation of the three vertices of that face. A minimum weight triangulation has good numerical properties and provides a close approximation of the function surface.

The complexity of the minimum weight triangulation problem has been open since 1975 when it was mentioned by Shamos and Hoey [18]. Since then, several algorithms have been proposed to solve this problem [12,16,18]. None of these is known to produce even a constant approximation of a minimum weight triangulation. On the other hand, though Lloyd [13] and Lingas [11] have proved the NP-completeness of related problems, efforts to show the minimum weight triangulation problem NP-hard have failed. In this paper we address this problem from two directions. We propose an algorithm that produces triangulations that are better approximations of a minimum weight triangulation than those produced by previous algorithms and also prove new NP-hardness results for two generalizations of the minimum weight triangulation problem.

We present an algorithm, called the greedy spanning tree triangulation algorithm (in brief, G-ST-T) that triangulates a set of n points in $O(n^3)$ time. G-ST-T has the flavor of a heuristic by Lingas [12], which we shall call the minimum spanning tree triangulation algorithm (briefly, MST-T). Both G-ST-T and MST-T follow a two step paradigm. In the first step, P is viewed as the vertices of a graph and $E(P)$ as its edges. Each algorithm choses a minimal subset $E^s(P)$ of pairwise non-crossing edges that span P. The choice of this set is crucial, and G-ST-T and MST-T differ in this choice. In the second step, an optimal triangulation containing $E^s(P)$ is obtained. optimally using a dynamic programming algorithm that is an extension of an algorithm due to Gilbert [5] that optimally triangulates the interior of a simple polygon. If every point of P is in the convex hull, then both algorithms optimally triangulate P. MST-T never produces a triangulation that has weight greater than that of the Delaunay triangulation, while G-ST-T never produces a triangulation that has weight greater than that of the greedy triangulation. This implies that the average case performance analysis of greedy algorithm due to Lingas [11] also applies to our algorithm. Experiments indicate that G-ST-T rarely produces a non-optimal triangulation and produces an optimal triangulation far more frequently than the greedy

algorithm. Even when G-ST-T fails to produce an optimal triangulation, its closeness to optimality is remarkable and certainly much better than any current algorithm.

We advance the notions of local optimality and k-optimality, and determine the local optimality of the known triangulation algorithms. We present an open question regarding the k-optimality of greedy triangulations, which if answered would provide further insights into the minimum weight triangulation problem.

We also present a formulation of the minimum weight triangulation problem as a graph problem. This leads to a variety of graph theoretic problems whose solution may have a bearing on the status of the minimum weight triangulation problem. We prove the NP-hardness of one such problem. Our other NP-hardness result is motivated by the NP-completeness result of Lloyd [13]. We show that given P, and $E^s(P)$, a subset of $E(P)$ that contains at least one triangulation of P, the problem of finding a minimum weight triangulation of P that is a subset of $E^s(P)$ is NP-hard.

The paper is organized as follows. In Section 2, we survey previous work related to the minimum weight triangulation problem. In Section 3, we discuss a dynamic programming algorithm that optimally triangulates the interior of a convex polygon in $O(n^3)$ time. We present an extension of this algorithm that optimally triangulates more general planar regions called "cells". In Section 4, we present Lingas' algorithm (MST-T) and our new algorithm (G-ST-T). Both algorithms run in $O(n^3)$ time. Though neither always produces an optimal triangulation, G-ST-T possesses properties that ensure that non-optimal triangulations are extremely rare. This is supported by experiments that compare the minimum weight triangulation with the performance G-ST-T, MST-T, and the greedy triangulation. These we present in Section 5. In Section 6, we investigate issues of local optimality related to the known triangulation algorithms. In Section 7, we present a graph theoretic formulation of the minimum weight triangulation problem and prove two new NP-hardness results. The final section of the paper, Section 8, contains some open problems and conjectures that arise from this work.

2 Previous approaches

Shamos and Hoey [18] first mention the minimum weight triangulation problem. They present a divide and conquer algorithm to construct a Voronoi diagram of n points in a plane in $O(n \log n)$ time. This implies that the Delaunay triangulation, which is the planar dual of the Voronoi diagram, can be constructed in $O(n \log n)$ time. A greedy triangulation is one that is produced by the greedy algorithm. The greedy algorithm always chooses the smallest edge not yet chosen that does not cross any previously chosen edge. Goldman [6] presents the most efficient known algorithm for producing a greedy triangulation; her algorithm runs in $O(n^2 \log n)$ time and $O(n)$ space. Shamos and Hoey [18] state that both the greedy and the Delaunay triangulations are optimal and hence the Delaunay algorithm is a more efficient way of computing a minimum weight triangulation than the greedy algorithm. Lloyd [13] provides counterexamples to show that both the Delaunay triangulation and the greedy triangulation are not always optimal. In fact, his counterexamples show that neither triangulation is optimal even for a convex set of points.

The complexity of the minimum weight triangulation is one of only four problems that remains open from Garey and Johnson's [4] original list of twelve open problems. In fact, there is no known polynomial time algorithm that produces a constant approximation of the minimum weight triangulation. Attempts to show the minimum weight triangulation problem NP-hard have resulted in two related NP-hardness results. In the earliest result, Lloyd [13] shows that given a set P of points in a plane and a subset E^s of $E(P)$, the problem of determining whether $E^s(P)$ contains even one triangulation is NP-complete. In a later result, Lingas [10] shows that the problem of determining the minimum weight geometric triangulation of multi-connected polygons is NP-complete.

The greedy and the Delaunay triangulations have been studied closely as approximations of minimum weight triangulation. That the greedy algorithm and the Delaunay algorithm do not produce an optimal triangulation is shown by the examples in Figures 2 and 3 respectively. Both examples are due to Lloyd [13] and consist of a set of vertices of a convex polygon. In the example of Figure 2 the greedy algorithm choses the line segments BD and AD for the triangulation, while a better triangulation is given by line segments AC and EC. In the example of Figure 3 the dashed lines constitute the Voronoi diagram and

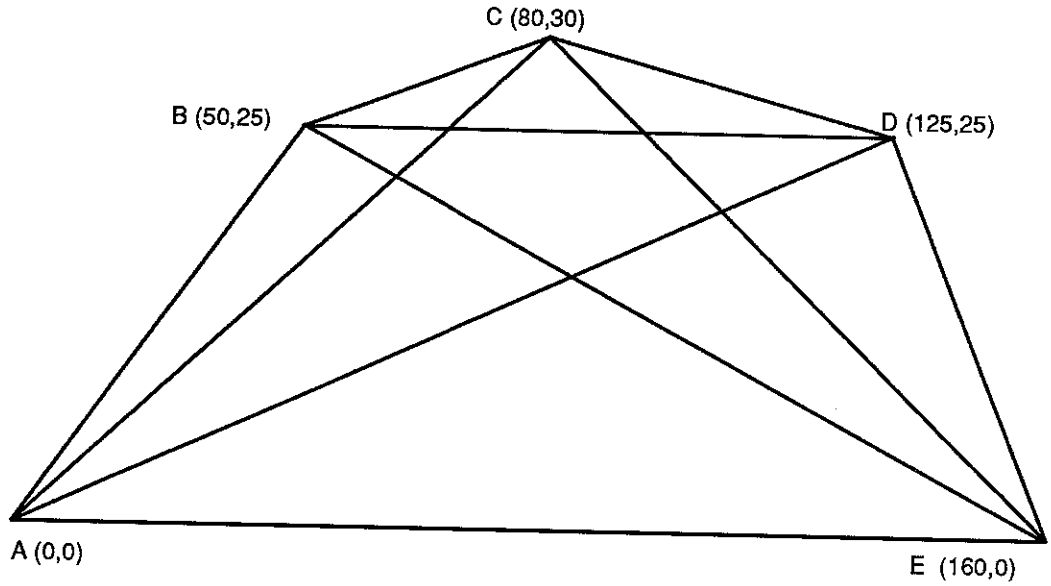


Figure 2: A counterexample to optimality of the greedy triangulation.

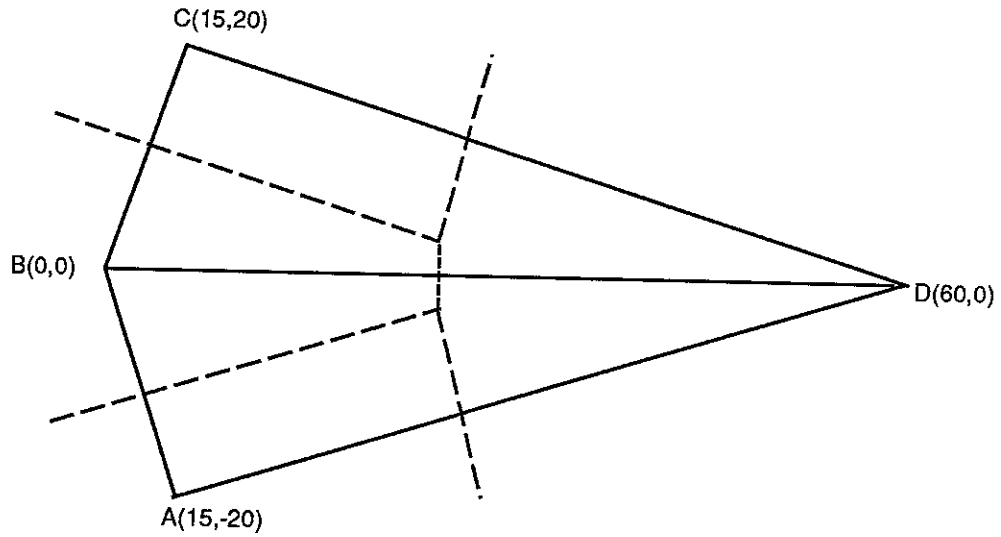


Figure 3: A counterexample to optimality of the Delaunay triangulation.

the solid lines constitute the Delaunay triangulation. The Delaunay triangulation contains line segment BD while an optimal triangulation is obtained by chosing line segment AC.

Let GT(P), DT(P), and MWT(P) denote a greedy triangulation, the Delaunay triangulation, and a minimum weight triangulation of P, respectively. A measure of how close a triangulation $T(P)$ is to a minimum weight triangulation is given by the ratio

$$R(T(P)) = \frac{W(T(P))}{W(\text{MWT}(P))}$$

Since neither the greedy triangulation nor the Delaunay triangulation are optimal, the worst-case ratios $R(\text{GT}(P))$ and $R(\text{DT}(P))$ give an indication of how well these triangulations approximate MWT(P). Manacher and Zobrist [14] construct sets of points P_0 and P_1, $|P_0| = |P_1| = n$ such that

$$R(\text{GT}(P_0)) = \Omega\left(n^{\frac{1}{3}}\right)$$

and

$$R(\text{DT}(P_1)) = \theta\left(\frac{n}{\log n}\right).$$

Levcopoulos [9] improves on the lower bound for the greedy triangulation by showing that for each $n > 4$, there exists a set P_2, $|P_2| = n$, such that

$$R(\text{GT}(P_2)) = \Omega\left(n^{\frac{1}{2}}\right).$$

These results imply that sets of points can be constructed for which the greedy triangulation and the Delaunay triangulation can be arbitrarily bad as compared to the minimum weight triangulation. Kirkpatrick [7] showed that for any triangulation $T(P)$, $R(T(P)) = O(n)$. Hence for the greedy triangulation there is a gap between the known upper bound of $O(n)$ and the lower bound of $\Omega(n^{1/2})$ as demonstrated by Levcopoulos. Kirkpatrick [7] demonstrated that for each n, a set of points P_3, $|P_3| = n$, can be constructed such that

$$R(\text{DT}(P_3)) = \theta(n).$$

This indicates that for certain sets of points, the Delaunay triangulation is as poor an approximation as possible.

Plaisted and Hong [16] present a triangulation algorithm and show that the weight of the triangulation that their algorithm produces is within $O(\log n)$ of the weight of the

optimal triangulation. Their implementation of the heuristic has a time complexity of $O(n^5)$. Smith [19] improves on this by implementing the algorithm of Plaisted and Hong in $O(n^2 \log n)$ time. The algorithm of Plaisted and Hong is in two steps. The first step takes as input a set P and produces a set of line segments that partitions the convex hull of P into convex polygons. The weight of the partition is the sum of the lengths of the convex hull and the interior line segments. Plaisted and Hong show that the weight of this partition is within a constant factor of the weight of an arbitrary partition of P into convex polygons. The second step triangulates each of the convex polygons produced in the first step. The weight of an optimal triangulation of an n point convex polygon is $O(\log n)$ times the perimeter of the polygon. This leads to the result that if PHT(P) is the triangulation produced by their algorithm, then $R(\text{PHT}(P)) = O(\log n)$. Plaisted and Hong use the ring heuristic [20] for their second step. The ring heuristic applied to a convex polygon produces a sequence of convex polygons $P_0, P_1 \ldots P_k$, where P_0 is the input convex polygon, P_k is a triangle and P_{i+1} is obtained from P_i by connecting alternate vertices of P_i. The union of these polygons is the triangulation that the ring heuristic produces. Plaisted and Hong conjecture that in fact $R(\text{PHT}(P)) = O(1)$. Olariu, Toida and Zubair [15] point out that this conjecture is false if the second step is performed using the ring heuristic. But this conjecture remains intact if the second step is performed using dynamic programming to optimally triangulate each convex polygon, though the time complexity of this step and hence of the whole algorithm will then be $O(n^3)$.

Lingas [12] presents an algorithm (MST-T) for the minimum weight triangulation problem that takes $O(n^3)$ time. The heuristic generates a polygon whose vertices are all the points in the input set. This polygon is the union of the convex hull and the minimum weight forest that connects all the interior points to the convex hull. The polygon is then triangulated optimally by dynamic programming. Lingas derives an upper bound that shows that

$$R(\text{MST-T}(P)) = O\left(\log n + \frac{n \times jump(P)}{W(\text{MWT}(P))}\right)$$

where MST-T(P) is the triangulation produced by his heuristic for a set of points P, and $jump(P)$ is the length of the longest line segment in the minimum weight spanning forest. Since $n \times jump(P)/W(\text{MWT}(P))$ can be as bad as $\Omega(n)$, this does not provide a non-trivial

worst case bound on the performance of this heuristic.

Using certain separator theorems, Smith [19] gives the first subexponential time algorithm to compute a minimum weight triangulation.

3 A Special Case

A special case of the triangulation problem arises when the points are the vertices of a convex polygon. Dynamic programming can be used to obtain an $O(n^3)$ solution to the problem [5]. The algorithm can be described as follows. Let $p_0, p_1, \cdots p_{n-1}$ be the n vertices of a convex polygon in clockwise order. Let $C[p_i, s]$, where $i = 0, 1, \ldots, n-1$ and $s = 1, 2, \ldots, n$, denote the cost of the interior edges of the minimum weight triangulation of the convex polygon $(p_i, p_{i+1} \cdots p_{i+s-1})$, where the subscripts are taken modulo n. The problem we wish to solve is that of determining $C[p_0, n]$, (or equivalently any problem $C[p_i, n]$, $i = 1, \ldots, n-1$). Then, $C[p_i, s] = 0$, $i = 0, 1, \ldots n-1$, $1 \leq s \leq 3$ and

$$C[p_i, s] = \min_{(i+1) \leq j \leq (i+s-2)} \{w(i,j) + w(j, i+s-1) + C[p_i, j-i+1] + C[p_j, i+s-j]\} \quad (1)$$

where $w(i,j) = 0$ if (p_i, p_j) is on the convex hull of $(p_0, p_1 \ldots p_{n-1})$. All additions and subtractions in the above equation are taken modulo n. Computing $C[p_i, s]$ can be thought of as filling in the entries of an $O(n^2)$ size table. Each computation of $C[p_i, s]$ takes $O(n)$ time and hence the entire algorithm runs in $O(n^3)$ time.

The problem of optimally triangulating a convex polygon in time less than $O(n^3)$ time is still open. Yao [22,23] presents a technique by which the time complexity of certain dynamic programming algorithms is reduced from $O(n^3)$ to $O(n^2)$. This technique requires the monotonicity of certain bivariate functions. For example, Equation (1) can be rewritten as

$$C[p_i, p_j] = \min_{i<k<j} \{w(i,k) + w(k,j) + C[p_i, p_k] + C[p_k, p_j]\} \quad (2)$$

where $C[p_i, p_j]$ is the cost of optimally triangulating the convex polygon $(p_i, p_{i+1} \ldots p_j)$. Using Equation (2), the following bivariate function can be defined

$$K(i,j) = \min\{k \mid w(i,k) + w(k,j) + C[p_i, p_k] + C[p_k, p_j]\}$$

If $K(i,j)$ is monotonic in i and j, i.e., if $K(i,j) \leq K(i,j+1) \leq K(i+1, j+1)$, then filling in the $O(n^2)$ entries of the cost table can be accomplished in $O(n^2)$ time instead of

$O(n^3)$ time. But counterexamples can be easily constructed to show that $K(i,j)$ is not monotonic in i and j for the minimum weight triangulation of convex polygons, and hence Yao's technique does not directly apply to this algorithm.

Note that the dynamic programming algorithm does not use the geometry of the problem. This indicates that a more general problem, which does not involve any geometry, can also be solved in $O(n^3)$ time. To state the problem we need some definitions. Let $P = \{p_0, p_1, \ldots, p_{n-1}\}$ and let $E(P) = \{(p_i, p_j) : p_i, p_j \in V, i < j\}$. Two elements, $(p_i, p_j) \in E(P)$ and $(p_k, p_l) \in E(P)$ intersect if $i < k < j < l$. To each element in $(p_i, p_j) \in E(P)$ we associate a positive real cost $c((p_i, p_j))$. The cost of a subset $E^s(P)$ is $c(E^s(P)) = \sum_{(p_i,p_j)\in E^s(P)} c(p_i, p_j)$. The problem can then be stated as

> Given P and $E(P)$, find a maximal subset of $E(P)$ of minimum cost that consists of pairwise non-intersecting elements.

Though we only presented the special case of a convex polygon, Gilbert's dynamic programming algorithm [5] optimally triangulates the interior of a simple polygon. We extend this algorithm to triangulate a more general figure that we call a *cell*. A cell is any interior face of a straight line planar embedding of a graph. Any cell can be uniquely represented by a sequence of vertices $p_0, p_1, \ldots, p_{n-1}$ encountered if the boundary of the cell is traversed in, say, a counterclockwise order. Note that the vertices in the sequence are not, in general, distinct and that there may be edges that are traversed twice. A cell with the points labeled is shown in Figure 4. It can be represented completely by the sequence $1, 2, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 10, 9, 12, 9, 8$. For the purposes of the algorithm every occurrence of the same point in the sequence is treated as being different. The dynamic programming algorithm that optimally triangulates the interior of a cell is similar to the one that optimally triangulates the interior of a convex polygon except that now the only line segments that can be considered are those that do not cross any line segments of the cell. This can be accomplished by assigning a weight of $+\infty$ to any line segment that crosses a line segment of the cell. Hence, we have the proposition

Proposition 3.1 *A cell of n points can be optimally triangulated in $O(n^3)$ time.*

In the next section we employ the above algorithm, which we call **Optimal cell triangu-**

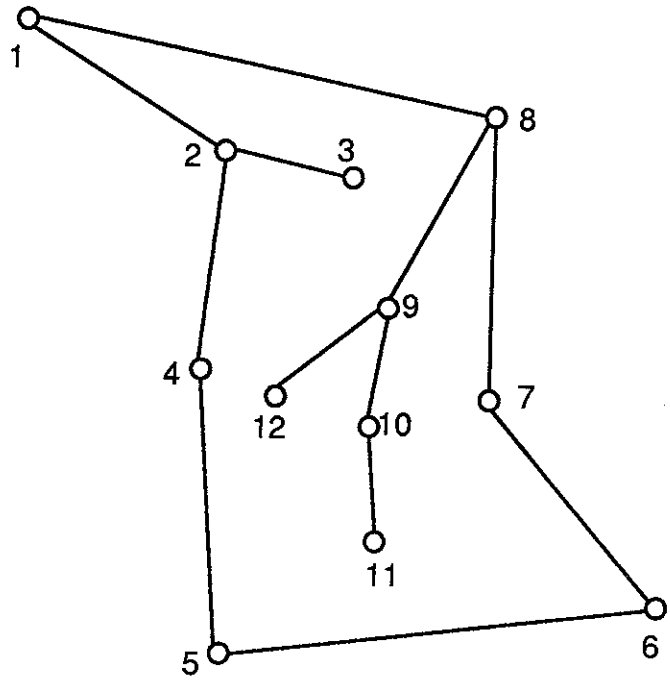


Figure 4: A 12 point cell.

lation (OCT).

4 An improved triangulation algorithm

We propose an algorithm (G-ST-T) that triangulates a set $P = \{p_1, p_2, \ldots, p_n\}$ of points in a plane in $O(n^3)$ time. G-ST-T has two distinct phases. In the first phase a spanning tree of the graph $(P, \mathrm{GT}(P))$ is chosen. How the spanning tree is chosen is explained later. The spanning tree along with the convex hull produce a single cell. In the second phase, the cell obtained in the first phase is optimally triangulated using the OCT algorithm. G-ST-T is similar in structure to the algorithm (MST-T) proposed by Lingas [12]. In fact, a similar analysis applies to G-ST-T also, and we include MST-T in our experimental analysis.

To introduce the two phase paradigm that is common to both MST-T and G-ST-T, we first present the minimum spanning tree triangulation algorithm.

Minimum spanning tree triangulation algorithm (MST-T)

Input : $P = \{p_1, p_2, \ldots, p_n\}$.

Output : A triangulation MST-T(P).

1. Compute the Delaunay triangulation DT(P).

2. To each line segment $(p_i,p_j) \in E(P)$ assign a cost as follows.

$$c((p_i,p_j)) = \begin{cases} 0 & \text{if } (p_i,p_j) \in CH(P) \\ w(p_i,p_j) & \text{otherwise} \end{cases}$$

   Utilizing DT(P), compute a minimum spanning tree $MST(P)$ of $(P,E(P))$ using the costs defined above.

3. Use OCT to optimally triangulate the single cell defined by $CH(P) \cup MST(P)$, producing MST-T(P).

The effect of the weights used in Step 2 is that all but one of the line segments in $CH(P)$ is in $MST(P)$. Steps 1 and 2 can each be performed in $O(n \log n)$ time while step 3 can be performed in $O(n^3)$ time. An example of a cell produced at the end of Step 2 in the MST-T algorithm is shown in Figure 5. The dark line segments constitute the minimum spanning tree chosen in step 2.

Lingas provides an upper bound on the performance of his algorithm as compared to the minimum weight triangulation. Let $MST(P)$ be a minimum spanning tree of the graph $(P,E(P))$. Let $I(P) = MST(P) - CH(P)$ be the interior line segments of the minimum spanning tree and let $jump(P) = \max\{w(p_i,p_j) : (p_i,p_j) \in I(P)\}$. For a triangulation $T(P)$ and a line segment l, let $A(T(P),l)$ be a set of line segments l' such that l' crosses l and at least one of the two pieces of l' between the crossing point and the endpoint of l' is not crossed by any other line segments in $I(P)$. The bound on the performance of MST-T follows from the following proposition that Lingas[12] proves.

Proposition 4.1 (Lingas[12]) *Let $T(P)$ be a triangulation of P. There exists a triangulation of the interior of the cell $CH(P) \cup MST(P)$ of weight at most*

$$6 \cdot \sum_{e \in I(P)} |A(T(P),e)| \times w(e) + (3 \log n + 9) \times W(T(P)) + (3 \log n + 2) \times W(I(P)).$$

This leads to the following bound on the weight of the triangulation produced by the MST-T algorithm

$$\frac{W(\text{MST-T}(P))}{W(\text{MWT}(P))} = O\left(\frac{n \times jump(P)}{W(\text{MWT}(P))} + \log n\right).$$

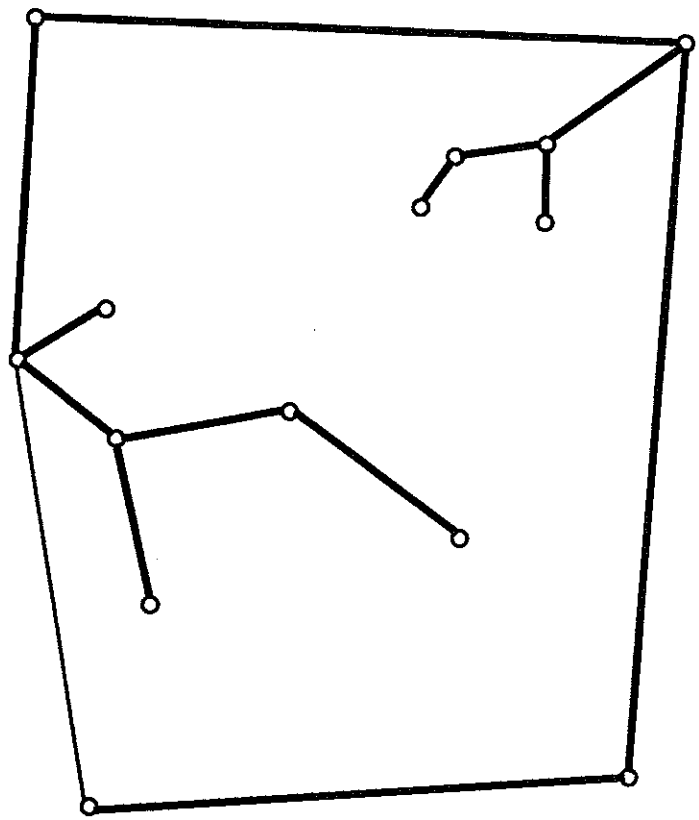


Figure 5: A simple polygon created at the end of step 2 in the MST-T algorithm.

Lingas gives an example for which $n \times jump(P)/W(\text{MWT}(P)) = \Omega(n)$ and yet the minimum weight triangulation is exactly the same as the triangulation produced by the MST-T algorithm. This shows that the bound is quite loose.

The following propositions follow immediately from the MST-T algorithm.

Proposition 4.2 *If P contains only vertices of a convex polygon, then* MST-T(P) *is optimal.*

Proposition 4.3 *For any set of points P,* MST-T(P) *is never worse than the Delaunay triangulation of P, i.e., $W(\text{MST-T}(P)) \leq W(\text{DT}(P))$.*

The MST-T algorithm will produce an optimal triangulation if and only if all the line segments in the spanning tree that we chose are also in some particular minimum weight triangulation. That this is not always the case is shown in Figure 6. Here, three of the four

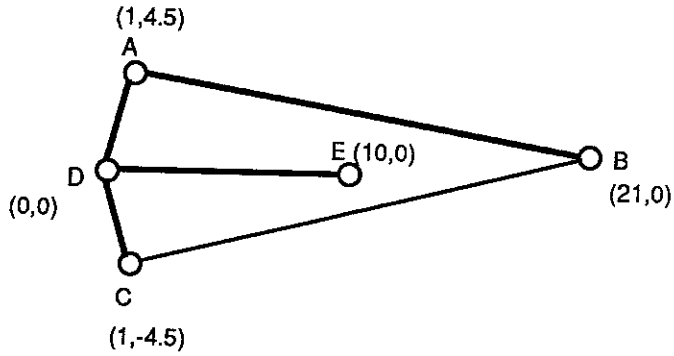


Figure 6: Counterexample to the MST-T algorithm.

convex hull edges are chosen to be in the minimum spanning tree. Whatever the choice may be, AC cannot be in the minimum spanning tree because it creates a cycle. Since, $w(DE) < w(BE) < w(AE) = w(CE)$, ED is the next edge chosen to be in the minimum spanning tree. The triangulation thus computed by MST-T algorithm is obviously non-optimal, since the optimal triangulation can be obtained by choosing edges AC, AE, CE and BE along with the convex hull edges.

The MST-T algorithm does not produce a non-optimal triangulation for any of the counterexamples in the literature [13,14] that have been used to show either the greedy triangulation or the Delaunay triangulation to be non-optimal. This can be attributed to the fact that the algorithm picks the least number of edges needed to keep the set of input vertices connected and then proceeds to select the optimal set of the remaining edges. So far we have not been able to say anything about the asymptotic behavior of the ratio R(MST-T(P)). We conjecture that R(MST-T(P)) = $O(1)$. This conjecture is motivated by the observation that the effects of a "bad" minimum spanning tree edge are local.

There are many ways of choosing a spanning tree in the first phase of the paradigm. The spanning tree chosen in the MST-T algorithm is one such choice. We now present an algorithm that differs from the MST-T algorithm in the choice of a spanning tree. Though the theoretical bound on the weight of the triangulation produced by our algorithm is loose, experimental results indicate that the choice of spanning tree that we prescribe produces

triangulations that are almost always optimal. The time complexity of this algorithm remains $O(n^3)$. We call this algorithm the *greedy spanning tree triangulation algorithm.*

Greedy spanning tree triangulation algorithm. (G-ST-T).

Input : $P = \{p_1, p_2, \ldots, p_n\}$.

Output : A triangulation G-ST-T(P).

1. Compute a greedy triangulation GT(P) of P and the convex hull $CH(P)$.
2. Compute a minimum spanning tree $GMST(P)$ of $(P, \mathrm{GT}(P))$ using the following assignment of costs to line segments

$$c((p_i,p_j)) = \begin{cases} \infty & \text{if } (p_i,p_j) \notin \mathrm{GT}(P) \\ 0 & \text{if } (p_i,p_j) \in CH(P) \\ w((p_i,p_j)) & \text{otherwise} \end{cases}$$

3. Use OCT to optimally triangulate the single cell in $CH(P) \cup GMST(P)$, producing G-ST-T(P).

An analysis of the time complexity of each step follows.

1. Preparata and Shamos [17] present an algorithm that produces a greedy triangulation. Their algorithm takes $O(n^2 \log n)$ time and $O(n^2)$ space. Goldman [6] presents a space efficient greedy triangulation algorithm that runs in $O(n^2 \log n)$ time and $O(n)$ space.
2. Since the number of edges in any planar graph is $O(n)$ the greedy triangulation contains $O(n)$ line segments. This implies that the second step can be accomplished in $O(n \log n)$ time.
3. As in the MST-T algorithm, the third step is performed in $O(n^3)$ time.

As before, the time complexity of the third step dominates and the time complexity of the algorithm is $O(n^3)$ time.

The following two propositions follow quite directly from the G-ST-T algorithm.

Proposition 4.4 *If P contains only vertices of a convex polygon, then* G-ST-T(P) *is optimal.*

Proposition 4.5 *For any set of points P,* G-ST-T(P) *is never worse than the greedy triangulation of P, i.e.,* $W(\text{G-ST-T}(P)) \leq W(\text{GT}(P))$.

Hence all upper bounds for the greedy triangulation also hold for triangulation produced by the G-ST-T algorithm. In particular all results about the asymptotic behavior of $R(\text{GT}(P))$ [8] are true for $R(\text{G-ST-T}(P))$. While minimal counterexamples for the greedy and the Delaunay triangulation require only five and four points respectively, any counterexample to G-ST-T must contain at least six points. This follows from the following theorem. In the proof we use the notation $\square ABCD$ to denote a quadrilateral with points A, B, C, and D in clockwise order.

Theorem 4.1 *There exists no five point set P for which* G-ST-T(P) *is not optimal, i.e., for which* $R(\text{G-ST-T}(P)) \neq 1$.

Proof: Let P be a set of five points. The configurations possible with five points are

1. $CH(P)$ contains three points.

2. $CH(P)$ contains four points.

We need not consider the case where all five points lie on the convex hull because of Proposition 4.4. The two cases are shown in Figure 7.

First consider Case 1. Line segment DE will be in any triangulation $T(P)$, because no line segment crosses DE. Of the quadrilaterals that line segment DE forms with the line segments of the triangle ABC, only one can be convex. (In Figure 7(a) the 6 quadrilaterals formed are $\square ADEC$, $\square AEDC$, $\square CEDB$, $\square CDEB$, $\square BDEA$, and $\square BEDA$). Without loss of generality, let the convex quadrilateral be $\square ADEC$. The diagonals AE and DC of $\square ADEC$ cross, and hence only one of them can be in any triangulation. None of the remaining line segments mutually cross, and hence they are in every triangulation. Since both the greedy triangulation and the minimum weight triangulation contain the smaller of line segments AE and DC we have that $\text{GT}(P) = \text{MWT}(P)$, from which it follows that $\text{G-ST-T}(P) = \text{MWT}(P)$.

Now consider Case 2 (Figure 7(b)). Suppose the diagonals AC and BD cross at a point O. The diagonals partition $\square ABCD$ into 4 quadrants, and point E lies in one of

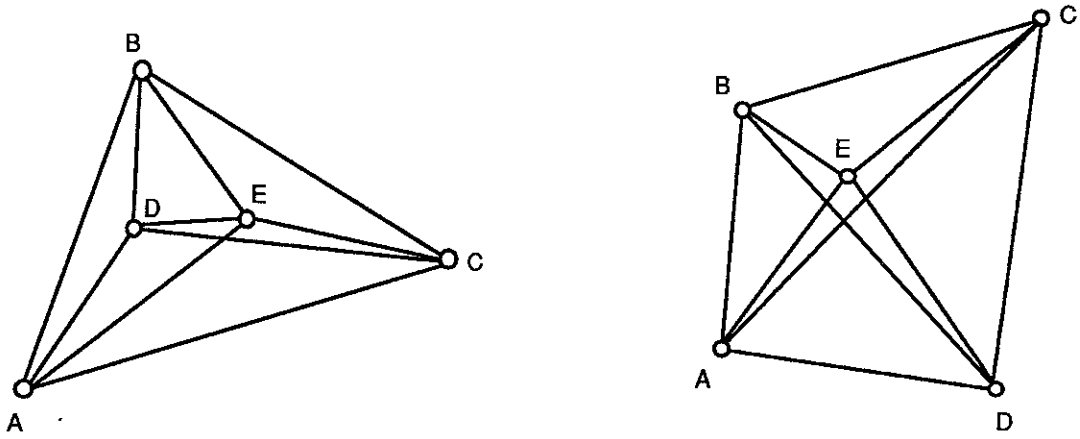
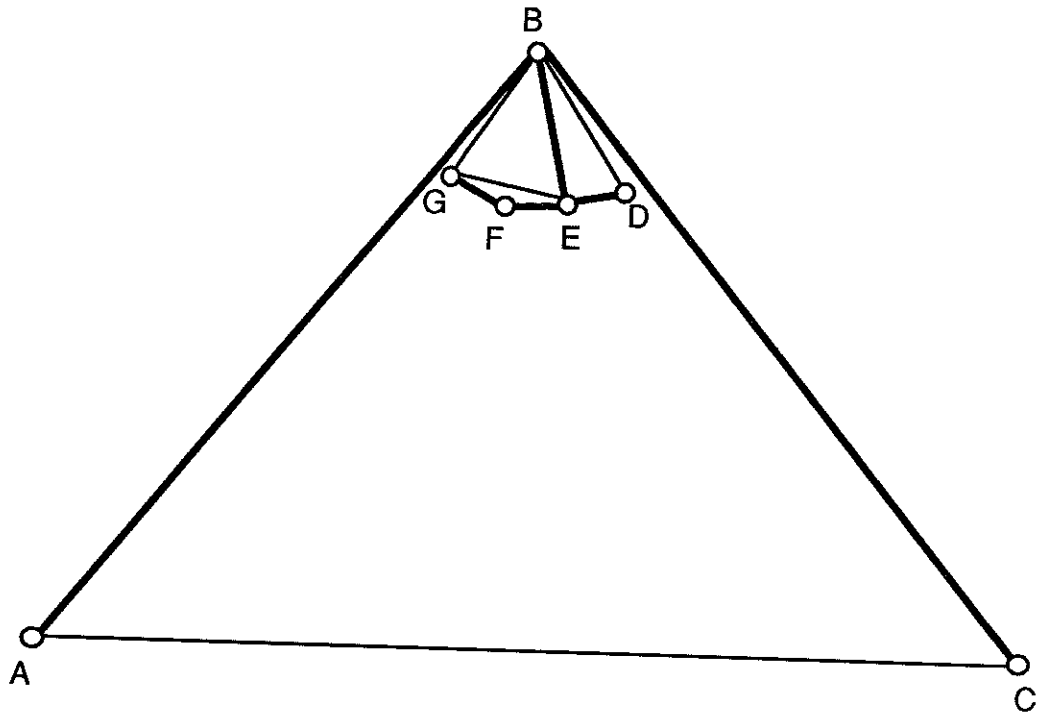


Figure 7: The two possible configurations with 5 points.

them. Without loss of generality, we may assume that E lies in quadrant BOC. Hence line segments BE and CE will be members of every triangulation. A minimum spanning tree of the greedy triangulation contains one of AE, BE, CE or DE. If this spanning tree contains either BE or CE then the triangulation produced by the G-ST-T algorithm is optimal. Consider the case where the greedy spanning tree contains either AE or DE. Without loss of generality, we may assume that the greedy spanning tree contains AE. Let L be the smaller of the two line segments DE and AC. The triangulation produced by the G-ST-T algorithm is $CH(\{A,B,C,D,E\}) \cup \{AE,BE,CE,L\}$. If the optimal triangulation contains AE then it is the same as the greedy spanning tree triangulation. Hence consider the case where the optimal triangulation contains line segment BD. Here the optimal triangulation is $CH\{A,B,C,D,E\}) \cup \{BD,BE,CE,DE\}$. But, $w(AE) \leq w(BD)$ since AE and BD cross and AE is in the greedy triangulation. Also $w(L) \leq DE$. Hence, the weight of the greedy spanning tree triangulation is no greater than the weight of the minimum weight triangulation. □

The G-ST-T algorithm will produce an optimal triangulation if and only if the line segments of the minimum spanning tree of the greedy triangulation are always in some particular minimum weight triangulation. That this is not always the case can be seen



B : (217,256). |BG| = 48.38, |BD| = 49.49,
D : (210,207). |BE| = 48.30, |BF| = 47.80,
E : (195,213). |GD| = 51.74,
F : (179,227). |GE| = 36.80, |FD| = 36.89.
G : (171,241).

Figure 8: Counterexample to the G-ST-T algorithm.

from the seven point counterexample in Figure 8. Here, GT(P) contains line segments BG, BD, GF, FE, ED, GE and BE (among others). The minimum spanning tree chosen from the line segments of GT(P) contains GF, FE, ED and BE (as well as two convex hull line segments). The triangulation that the G-ST-T algorithm computes is the same as the greedy triangulation. A slight improvement in the weight of the triangulation can be obtained by choosing line segments FD and BF instead of GE and BE. Examples for which both the MST-T and the G-ST-T algorithms produce non-optimal triangulations can be constructed by combining the examples in Figure 6 and Figure 8. G-ST-T produces an optimal triangulation for the example in Figure 6, while MST-T produces an optimal triangulation for the example in Figure 8, thus showing that G-ST-T and MST-T are not

Set Size	Number of failures		
	Greedy	MST-T	G-ST-T
15	44	5	1
20	61	11	0
25	69	12	1
Total	174	28	2

Table 1

comparable as the G-ST-T and the greedy algorithm are.

Proposition 4.1 can be used to construct an upper bound on the size of the triangulation produced by the G-ST-T algorithm. For this we define $greedy\_jump(P)$ as the largest line segment in $GMST(P) - CH(P)$. Then we have the following theorem.

Theorem 4.2 *For any set of points P,*

$$R(\text{G-ST-T}(P)) = O\left(\frac{n \times greedy\_jump(P)}{W(\text{MWT}(P))} + \log n)\right).$$

It is clear that for any set of points P, $greedy\_jump(P) \geq jump(P)$. If for all sets of points P, $greedy\_jump(P) = O(jump(P))$, then the bound specified in proposition 4.1 also holds for the G-ST-T algorithm.

5 Experimental results.

The greedy, MST-T, and G-ST-T algorithms were tested with sets of sizes 15, 20 and 25 with 400 sets of each size. Testing larger sets was impractical due to the time required by the exhaustive search for finding a minimum weight triangulation algorithm. The points in each set were generated randomly with integer coordinates between 0 and $2^{16} - 1$. The performance of these algorithms was compared to the weight of the optimal triangulation, which was computed through an exhaustive search of the space of triangulations. Three kinds of comparisons were made.

Table 1 lists, for each set size, the number of times each algorithm fails to produce an optimal triangulation for that set size. It should be noted that the G-ST-T algorithm failed to produce the optimal triangulation only twice in 1200 trials as compared to the greedy algorithm which failed 174 times. The MST-T algorithm also does quite well as compared

Set Size	Average triangulation weight			
	Optimal	Greedy	MST-T	G-ST-T
15	298697.76	309873.34	299236.84	298697.76
20	349238.62	373288.61	350176.36	349238.62
25	437233.88	479283.36	439846.97	437233.88

Table 2

Set Size	Worst case % over optimal		
	Greedy	MST-T	G-ST-T
15	4.4963	0.9701	0.0017
20	10.1832	1.0017	0.0000
25	17.3897	1.5323	0.0003

Table 3

to the greedy algorithm, failing 28 times in 1200 trials. These results are evidence that MST-T and G-ST-T rarely produce non-optimal triangulations.

Table 2 is a comparison of the average size of the triangulations produced by each algorithm for each set size. G-ST-T fails only once each for set size 15 and set size 25, and never for set size 20. Even when the algorithm fails, the weights of the triangulations produced by G-ST-T algorithm are very close to the minimum weight triangulation. These results are evidence that in the average case G-ST-T and MST-T produce triangulations that are within a small constant of approximation of the minimum weight triangulation.

Table 3 compares the worst case triangulation produced by each algorithm. For each triangulation $T(P)$ and for each set size the worst case ratio

$$\frac{W(T(P)) - W(\text{MWT}(P))}{W(\text{MWT}(P)).}$$

is shown in Table 3. The results in this table are evidence that in the worst case both MST-T and G-ST-T algorithms produce triangulations that are within $O(1)$ of the minimum weight triangulation.

6 Issues of Local optimality

Throughout the paper we have been calling a minimum weight triangulation "optimal". We will now elaborate on the notion of optimality. Given a set of points P and a triangulation $T(P)$ of P, we say that $T(P)$ is *locally optimal* if and only if every convex quadrilateral in $T(P)$ is optimally triangulated.

It follows from the definition that for every convex quadrilateral Q in a locally optimal triangulation $T(P)$, the shorter diagonal of Q belongs to $T(P)$. While a minimum weight triangulation is always locally optimal, the converse does not always hold. The status of the four triangulations discussed in this paper with regard to local optimality is presented in the following theorem.

Theorem 6.1 *With local optimality as defined above, we have that*

1. *A greedy triangulation is always locally optimal.*
2. *A Delaunay triangulation is not always locally optimal.*
3. *A minimum spanning tree triangulation is not always locally optimal.*
4. *A greedy spanning tree triangulation is not always locally optimal.*

Proof: From the definition of a greedy triangulation, it follows immediately that the greedy triangulation is locally optimal. An example showing that the Delaunay triangulation is not always locally optimal can be found in Figure 3. In this example, $\square BCDA$ is a convex quadrilateral in the triangulation that is not optimally triangulated. An example showing that the MST-T algorithm does not always produce a locally optimal triangulation can be found in Figure 6. In this example, $\square DAEC$ is a convex quadrilateral in the triangulation that is not optimally triangulated. That the G-ST-T is also not always locally optimal is a surprising result, and the example in Figure 9 shows this. In this example the points A, B, C, D, E, and F are in the interior of a large convex polygon. In the upper figure the greedy triangulation of these points is shown with the minimum spanning tree indicated by the darkened line segments. In the lower figure the output of G-ST-T is shown. The two triangulations differ in that, the greedy triangulation contains line segments DF and DB while the triangulation produced by G-ST-T contains line segments AE and BE. AB is in

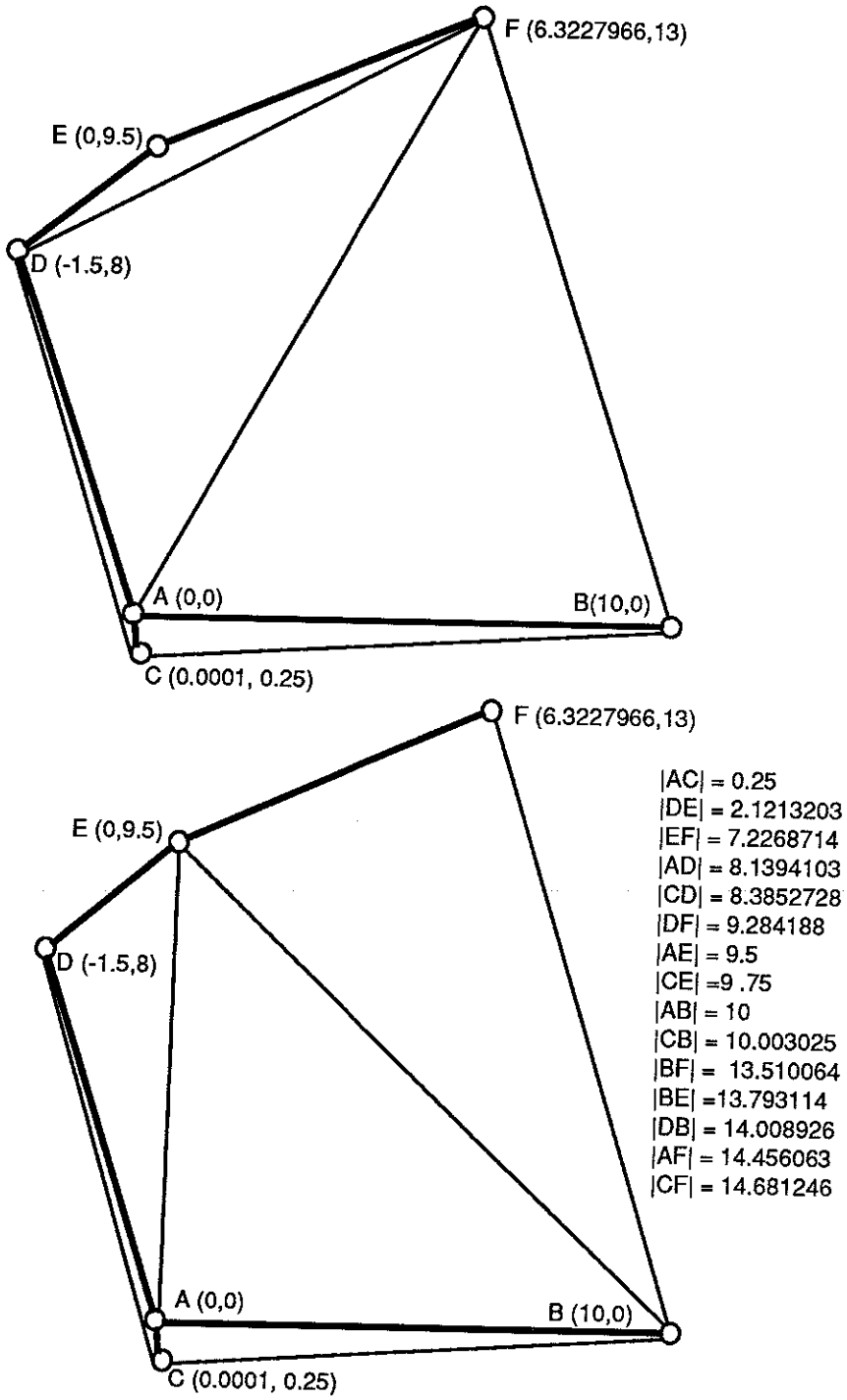


Figure 9: An example to show that G-ST-T algorithm does not always produce locally optimal triangulations.

the greedy triangulation and also in the minimum spanning tree of the greedy triangulation. In the output of G-ST-T AB is the diagonal of quadrilateral $AEBC$ and diagonal AB is longer than diagonal EC. □

The really surprising result of Theorem 3 is that a triangulation produced by G-ST-T is not always locally optimal. The G-ST-T triangulation is derived from the locally optimal greedy triangulation, yet fails to inherit the local optimality.

We extend the notion of local optimality and define the notion of k-optimality in the following manner. A triangulation $T(P)$ is *k-optimal*, $k \geq 3$, if every k vertex convex polygon belonging to $T(P)$ is optimally triangulated.

By this definition any triangulation is 3-optimal, while any locally optimal triangulation is 4-optimal. Observe that if a triangulation is k-optimal, $k \geq 4$, then the triangulation is also $(k-1)$-optimal. A triangulation is *k-non-optimal* if it is not k-optimal. A triangulation is *strictly k-non-optimal* if it is not k-optimal, yet it is j-optimal for all $j \leq k$. An interesting question regarding the local optimality of the greedy triangulation is the following:

> Is there a constant N such that there is no set of points P, for which GT(P) is strictly i-non-optimal, for all $i \geq N$?

The same question can be restated as

> Is there a constant N such that for any set of points P, if $GT(P)$ is k-non-optimal, where $k \geq N$ then $GT(P)$ is j-non-optimal for some $j < k$?

If there were a constant N satisfying the above question, then a minimum weight triangulation can be constructed by first constructing a greedy triangulation and then searching for every convex polygon of size $N-1$ or less, that is not optimally triangulated. Every such convex polygon is retriangulated using dynamic programming to yield a minimum weight triangulation. The worst case time complexity of this algorithm will depend on the number of non-optimal convex polygons that are created each time a non-optimal convex polygon is retriangulated. We shall call the constant N, the threshold of strict local non-optimality and prove a surprising lower bound for it.

Theorem 6.2 *If N is the threshold of strict local non-optimality then $N > 5$.*

Proof: We know that $N > 4$ because Figure 2 gives an example of a pentagon that is not optimally triangulated, but all quadrilaterals are. We also present an example with six points for which the greedy triangulation contains a non-optimal hexagon, yet all pentagons in the triangulation are optimally triangulated. This is shown in Figure 10. Here the hexagon $ABECDF$ is not optimally triangulated but the two pentagons $ABCDF$ and $ABECD$ are optimally triangulated in the greedy triangulation. □

The answer to the above would be very interesting in its own right. As k increases, maintaining strict k-non-optimality imposes more and more constraints on the points and we believe that for some k, meeting all the constraints would be geometrically impossible and hence we conjecture the existence of the constant N.

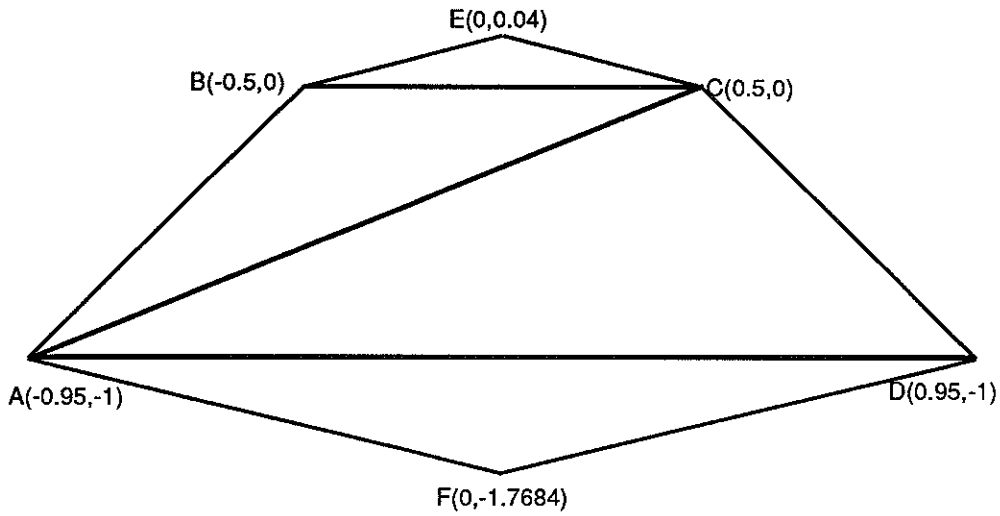
7 NP-hardness results

In keeping with the notation of Garey and Johnson, the minimum weight triangulation problem can be stated as a decision problem as follows

MINIMUM WEIGHT TRIANGULATION (MWT)
INSTANCE. A set P of points in a plane, a positive rational number k.
QUESTION: Is there a triangulation of P whose weight is no more than k?

The computational complexity of this problem remains open. The minimum weight triangulation problem can be viewed in various ways. A particular view is to consider P as the set of vertices of a graph G and $E(P)$ as the set of edges of G. G is the complete graph on $|P|$ vertices. The lengths of the line segments in $E(P)$ can be thought of as weights associated with the edges of G. This graph $G = (P, E(P))$ made from the points and line segments has an obvious straight line embedding in the plane. Let $C = \{(e_i, e_j) \in E(P) \times E(P) : e_i \text{ and } e_j \text{ cross }\}$ be the set of crossings in the embedding. Then $G^c = (E(P), C)$ is the *crossing graph* of G. Each vertex (line segment) of G^c has weight equal to its Euclidean length.

Figure 11 illustrates a graph G and its crossing graph G^c. The minimum weight triangulation problem can be posed in terms of the crossing graph G^c as follows:



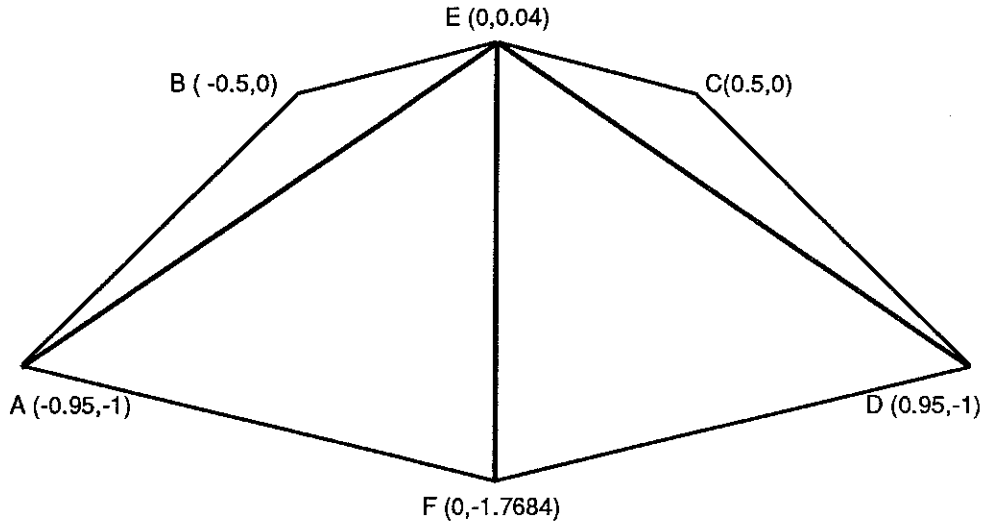


Figure 10: A greedy triangulation with a non-optimally triangulated hexagon and with all pentagons optimally triangulated.

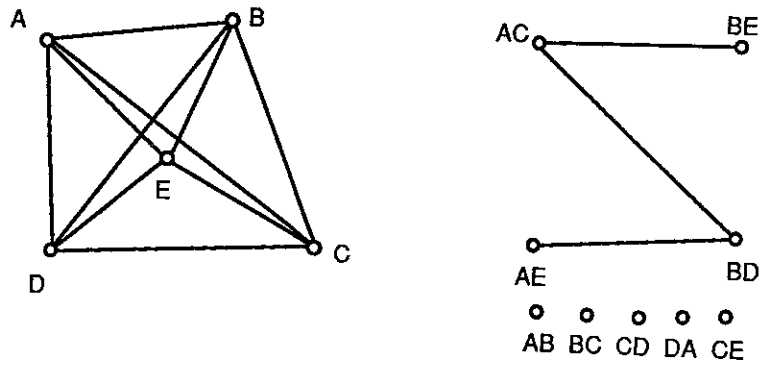


Figure 11: A complete straight line graph and its crossing graph.

Given a crossing graph, G^c, find a maximal independent vertex set of minimum total weight.

The problem can be restated as a decision problem as follows:

RESTRICTED MINIMUM MAXIMAL INDEPENDENT SET

INSTANCE. A crossing graph $G^c = (E, C)$, positive rational weight $w(e)$ for each $e \in E$, and a positive rational number k.

QUESTION: Is there a maximal independent set I in G^c such that $\sum_{e \in I} w(e) \leq k$.

If we discard the restriction that the graph under consideration is a crossing graph, then we have a more general problem.

MINIMUM MAXIMAL INDEPENDENT SET

INSTANCE. A vertex weighted graph $G = (V, E)$, positive rational weight $w(v)$ for each vertex $v \in V$, and a positive rational number k.

QUESTION: Is there a maximal independent set I, such that $\sum_{v \in I} w(v) \leq k$?

We show that the above problem is NP-hard. The reduction is in two steps and it makes use of the following problems.

SMALLEST MAXIMAL INDEPENDENT SET

INSTANCE. A graph $G = (V, E)$, integer $k \geq 0$.

QUESTION: Is there a maximal independent set $I \in V$ such that $|I| \leq k$?

MINIMUM MAXIMAL MATCHING

INSTANCE. A graph $G = (V, E)$, integer $k \geq 0$.

QUESTION: Is there a maximal matching M in G such that $|M| \leq k$?

Yannakakis and Gavril [21] show that MINIMUM MAXIMAL MATCHING is NP-complete. We now show that MINIMUM MAXIMAL INDEPENDENT SET is NP-hard.

Theorem 7.1 *MINIMUM MAXIMAL INDEPENDENT SET is NP-hard.*

Proof: It is obvious that if SMALLEST MAXIMAL INDEPENDENT SET is NP-hard, then so is MINIMUM MAXIMAL INDEPENDENT SET. We show that SMALLEST MAXIMAL INDEPENDENT SET is NP-hard by reduction from MINIMUM MAXIMAL MATCHING. Let $G = (V, E)$ and an integer $k \geq 0$, be an instance of MINIMUM MAXIMAL MATCHING. Then an instance of SMALLEST MAXIMAL INDEPENDENT SET is $G^l = (V^l, E^l)$ and k, where G^l is the line graph of G. G contains a maximal matching M, such that $|M| \leq k$, if and only if G^l contains a maximal independent set I such that $|I| \leq k$. □

If P is the set of vertices of a convex polygon, then the complete straight edge graph induced by these vertices $G = (P, E(P))$ can be embedded in a circle with $O(n^2)$ chords while maintaining all crossings. The crossing graph of G is, therefore, a circle graph with $O(n^2)$ vertices. Buckingham [2] presents an algorithm to determine the maximal independent set with the maximum weight in a weighted circle graph. The algorithm assumes the following labeling of the endpoints of the chords. The labeling begins at an arbitrary point on the circle and proceeds in the clockwise direction. The first endpoint encountered is labeled 1 and its paired endpoint is labeled $1'$. If i chords are labeled, then the next endpoint encountered is labeled $(i+1)$ and its paired endpoint labeled $(i+1)'$. This labeling can be seen in Figure 12. Corresponding to such a labeling of the endpoints of the chords we can say that certain chords are *contained* in certain other chords. A chord (i, i') is contained in a chord (j, j') if, when traversing the circle in clockwise direction the order in which the four endpoints occur is (j, i, i', j'). The algorithm presented in [Buc80] takes $O(c+n)$ time , where n is the number of vertices in the circle graph and c is the number of instances of a

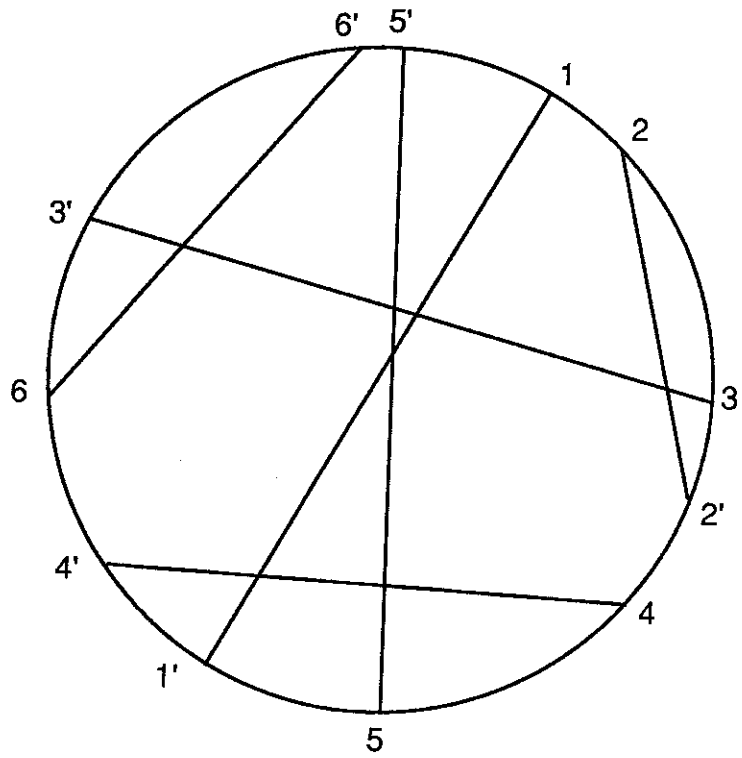


Figure 12: The labeling of the endpoints of the chords of a circle.

chord being strictly contained within another chord. This algorithm can be modified easily to obtain an algorithm that determines the maximal independent set with the minimum weight in $O(c+n)$ time. It is clear that G has $O(n^2)$ chords and that $c = O(n^3)$. Hence, this can be thought of as an alternate $O(n^3)$ time algorithm that optimally triangulates a set of points which are the vertices of a convex polygon. While this is not an improvement over the existing best time complexity for this problem, this may provide insights into ways to improve on this time complexity.

The second NP-hardness result that we prove is for a geometric decision problem. This problem is a generalization of MWT. Lloyd proved the following problem NP-complete. In keeping with Lloyd's notation we denote $E(P)$ by E.

TRIANGULATION EXISTENCE (TRI)
INSTANCE: A set P of points in a plane, a set of line segments $E' \subseteq E$.
QUESTION: Does E' contain a triangulation of P?

This problem was shown to be NP-complete by a reduction from SAT. Motivated by the above problem, we propose the following generalization of MWT.

GENERALIZED MINIMUM WEIGHT TRIANGULATION (GMWT)
INSTANCE: A set P of points in a plane, a set of line segments $E' \subseteq E$, such that E' contains a triangulation of P, a positive rational number k.
QUESTION: Is there a triangulation in E' whose weight is no more than k?

Theorem 7.2 *GMWT is NP-hard.*

Proof: GMWT is shown to be NP-hard by a reduction from SAT. This reduction is a modification of the reduction that Lloyd [13] used to show TRI NP-complete and hence after an overview we only mention the modifications to Lloyds proof.

Assume that we have clauses $C_1, C_2, \ldots, C_k$ each of which is a disjunction of literals drawn from the variables $x_1, x_2, \ldots, x_n$. The building block of the construction is a *switch* S_{ij} corresponding to each clause, literal pair. The instance of GMWT corresponding to the instance of SAT is a rectangular array of switches called a *network*. The switches are

labeled as points with Cartesian coordinates with the indices of the variables on the x-axis and the indices of the clauses on the y-axis. The switches are of three types : S_{ij} is positive if $x_i \in C_j$, S_{ij} is negative if $\overline{x_i} \in C_j$ and S_{ij} is neutral if $x_i \notin C_j$ and $\overline{x_i} \notin C_j$. In any triangulation of this network we can think of streams of current flowing in two directions. A vertical stream of current flowing upwards represents the truth value of variable while a horizontal stream of current flowing to the right represents the evaluation of the truth value of a clause. For each variable x_i, the same truth value flows through each switch S_{ij}, $1 \leq j \leq k$. The construction forces the horizontal current flowing into each switch S_{1j} to be false while the horizontal current flowing out of each switch S_{nj} can be true or false. Lloyd's construction forces the truth value flowing out of each switch S_{nj} to be true; our construction differs from his in this regard. A smaller triangulation corresponds to a horizontal current that changes truth value to true while a larger triangulation corresponds to a truth value that remains false. Hence, the smallest triangulation corresponds to every clause being satisfied.

A switch suitable for our reduction can be constructed from the one used by Lloyd by making the following changes to Lloyd's construction.

1. Each of the squares in the corners of a switch is shrunk, keeping the coordinates of points E_1, E_2, E_3 and E_4 the same. The octagon at the center of the switch is shrunk into a geometrically similar octagon that has the same center as the original octagon. The shrinking continues until each of the switches seems to contain 5 points, 4 at each of the corners of a square and 1 at the center. It can be verified that this shrinking preserves the pairwise intersection of line segments.

2. The coordinates of the special points V^{nj} are changed from $(100 \cdot n, 100 \cdot (j-1)+50)$ to $(x, 100 \cdot (j-1)+50)$. $x > 100 \cdot n$ is chosen such that point V^{nj} is "far enough" from points I^{nj} and H^{nj}. The notion of "far enough" is made precise later in the proof. Let the length of edge (V^{nj}, I^{nj}) and edge (V^{nj}, H^{nj}) be d.

3. For each switch S^{nj} we add a new point X^{nj} to the set of points V. We do not specify the coordinates of X^{nj} beyond saying that X^{nj} lies in the intersection of the interior of $\Delta A_1^{nj} H^{nj} I^{nj}$, $\Delta B_1^{nj} H^{nj} I^{nj}$, $\Delta C_1^{nj} H^{nj} I^{nj}$, $\Delta D_1^{nj} H^{nj} I^{nj}$ and $\Delta Q^{nj} H^{nj} V^{nj}$. Call

the new set of points V'.

4. For each switch S^{nj} add the following edges to E.

$$\{X^{nj}\} \times \{C_1^{nj}, D_1^{nj}, I^{nj}, H^{nj}, Q^{nj}, V^{nj}\} \qquad 1 \leq j \leq k$$

$$(I^{nj}, H^{nj}), (Q^{nj}, V^{nj}) \qquad 1 \leq j \leq k$$

$$(V^{nj}, V^{n(j+1)}) \qquad 1 \leq j < k.$$

Call the new set of edges E'.

To complete the instance of GMWT we let

$$M = (1600 \cdot \sqrt{2} + 100) \cdot n \cdot k + 350 \cdot k + 100 \cdot n + 2 \cdot d \cdot k.$$

The next step is to show that an instance of SAT is satisfiable if and only if an instance of GMWT constructed in the above manner contains a triangulation whose weight is no more than M.

If the given instance of SAT is satisfiable, than by Lloyd's proof it follows that there exists a triangulation T of V in E. A triangulation T of V implies that the east-connected point in switch S^{nj} is either C_1^{nj} or D_1^{nj}. A triangulation T' of V' can be constructed from T as follows:

$$\begin{aligned} T' = T \quad & \cup \quad \{(X^{nj}, I^{nj}), (X^{nj}, H^{nj}), (H^{nj}, I^{nj}) | 1 \leq j \leq k\} \\ & \cup \quad \{(X^{nj}, C_1^{nj}) | 1 \leq j \leq k \text{ and } C_1^{nj} \text{is exposed } \} \\ & \cup \quad \{(X^{nj}, D_1^{nj}) | 1 \leq j \leq k \text{ and } D_1^{nj} \text{is exposed}\} \\ & \cup \quad \{(V^{nj}, V^{n(j+1)}) | 1 \leq j < k\}. \end{aligned}$$

We now compute the weight of T' and show that it is no greater than M.

As Lloyd showed, in any triangulation of the network, a switch S^{ij} can be triangulated in one of four ways, called A-triangulation, B-triangulation, C-triangulation and D-triangulation. The edges in any triangulation of a switch are of four types. They are

1. corner edges : connect frame vertices to frame vertices in the same corner.

2. diagonal edges : connect frame vertices to frame vertices in the opposite corner.

3. half-diagonal edges : connect frame vertices to terminal vertices.

4. center edges : connect terminals.

The corner edges and the center edges have weights approximately equal to 0, while each diagonal edge has a weight approximately equal to $100 \cdot \sqrt{2}$ and each half-diagonal edge has weight approximately equal to $100/\sqrt{2}$. An A-triangulation and a D-triangulation of a switch contain 26 half-diagonals and 3 diagonals. A B-triangulation or a C-triangulation contain 24 half diagonals and 2 diagonals. Hence, the weight of an A-triangulation or a D-triangulation is $1600 \cdot \sqrt{2}$ and the weight of a B-triangulation or a C-triangulation is $1400 \cdot \sqrt{2}$. Hence, the weight of the largest triangulation of a switch is $1600 \cdot \sqrt{2}$. The weight of each inter-switch edge is approximately 100 and since X^{nj} can be made arbitrarily close to H^{nj}, the weight of the interior edges of the triangulation of $\Delta C_1^{nj} H^{nj} I^{nj}$ or $\Delta D_1^{nj} H^{nj} I^{nj}$ is approximately 150. The weight of the convex hull of V is $2 \cdot n \cdot 100 + 2 \cdot k \cdot 100$. The weight of the set of edges $\{(V^{nj}, V^{n(j+1)}) | 1 \leq j < k\}$ is $(k-1) \cdot 100$ and the weight of the edges (V^{nj}, I^{nj}) and (V^{nj}, H^{nj}) is d each. Hence the total weight of the triangulation T' is given by

$$\begin{aligned} W(T') &= 1600 \cdot \sqrt{2} \cdot n \cdot k + 100 \cdot (n-1) \cdot (k-1) + 150 \cdot k + 200 \cdot n + \\ &\quad 200 \cdot k + 100 \cdot (k-1) + 2 \cdot d \cdot k \\ &= (1600 \cdot \sqrt{2} + 100) \cdot n \cdot k + 350 \cdot k + 100 \cdot n + 2 \cdot d \cdot k. \end{aligned}$$

Since $W(T') = M$, we have shown that if SAT is satisfiable then there exists a triangulation T' of V' such that the weight of T' is no more than M.

To show the only if part, we assume that E' contains a triangulation T' of V' such that the weight of T' is no more than M. In triangulation T', if B_1^{nj} is exposed then since there is no edge (B_1^{nj}, X_1^{nj}) in E', the pentagon $(I^{nj}, B_1^{nj}, Q^{nj}, H^{nj}, V^{nj})$ containing X^{nj} is triangulated by the edges (Q^{nj}, V^{nj}), (B_1^{nj}, V^{nj}),(X^{nj}, Q^{nj}),(X^{nj}, H^{nj}) and (X^{nj}, V^{nj}). The weight of the interior edges of the triangulation of this pentagon is greater than $3 \cdot d$ and the weight of the triangulation of this switch is $3 \cdot d + 1350 \cdot \sqrt{2}$. The situation when A_1^{nj} is exposed is exactly the same. Hence, the weight of a triangulation T' which contains a switch S^{nj} with A_1^{nj} or B_1^{nj} exposed is greater than

$$1400 \cdot \sqrt{2} \cdot (n \cdot k - 1) + 100 \cdot (n-1) \cdot (k-1) + 100 \cdot k + 200 \cdot n +$$

$$(k-1)\cdot 100 + 2\cdot d\cdot k + 3\cdot d + 1350\cdot\sqrt{2}.$$

Let N denote the above number. By comparing M with N it can be seen that, if

$$3\cdot d + 1350\cdot\sqrt{2} > 1600\cdot\sqrt{2}\cdot n\cdot k$$

then N is greater than M. Hence, by choosing V^{nj} "far enough" such that d satisfies the above inequality, we ensure that if T' has size no greater than M, then for all switches S^{nj}, either vertex C_1^{nj} is exposed or vertex D_1^{nj} is exposed. This implies that E contains a triangulation T of V and by Lloyd's proof this implies that the corresponding instance of SAT is satisfiable. □

GMWT is the closest problem to MWT that has been shown to be NP-hard. Additional insights may allow the above proof to be extended to MWT.

8 Conclusions and Conjectures

We have made progress in two directions towards determining the complexity of the minimum weight triangulation problem. In the first direction, we present two algorithms, one of which is due to Lingas (MST-T) [12] which experimentally perform much better than previous algorithms. While MST-T never performs worse than the Delaunay triangulation, our algorithm (G-ST-T) never performs worse than the greedy triangulation. The bound proved by Lingas in proposition 4.4 will hold for G-ST-T provided $greedy\_jump(P) = O(jump(P))$ for any set of points P. We conjecture that this is true. Experimental results show that our algorithm rarely produces non-optimal triangulations as compared to the greedy triangulation and the triangulation produced by Lingas' algorithm. Even when a non-optimal triangulation is produced by our algorithm, it is very close in weight to the minimum weight triangulation. Both algorithms also produce a minimum weight triangulation when P only contains vertices of a convex polygon. These results lead us to the following conjecture

> **Conjecture 1:** MST-T and G-ST-T algorithms produce triangulations that are within $O(1)$ of the minimum weigh triangulation in the worst case.

Our algorithm also is a specific instance of a general strategy of choosing a small subgraph from the complete graph of line segments induced by a set of points P and then using

dynamic programming to optimally triangulate the interior of the straight line embedding of the subgraph. This strategy can be applied to Plaisted and Hong's triangulation algorithm also, to produce a triangulation which is always within $O(\log n)$ of the optimal and probably much better than the triangulation produced by their algorithm.

The OCT algorithm that we use in the G-ST-T algorithm, pays no attention to the geometry of the problem and hence solves a more general problem with the same time complexity. We conjecture that using the geometry will improve the time complexity of the OCT algorithm and hence also the time complexity of our algorithm. Even for special cases of a convex polygon such as a semicircular polygon, the best known algorithm that optimally triangulates the interior of the polygon, still takes $O(n^3)$ time. Hence, it is an open question whether there are some classes of convex polygons for which the dynamic programming algorithm can be improved from $O(n^3)$ time complexity.

There are many questions regarding the time complexity of algorithms that produce a greedy triangulation. Currently the best is an $O(n^2 \log n)$ time and $O(n)$ space algorithm by Goldman [6]. The greedy triangulation of a convex polygon can be computed in $O(n^2)$ time and the greedy triangulation of a semicircular polygon can be computed in $O(n \log n)$ time [11]. It is unknown whether these time complexities can be further improved. In the G-ST-T algorithm we compute a minimum spanning tree of a greedy triangulation by first constructing the greedy triangulation and then choosing a minimum spanning tree. This takes $O(n^2 \log n)$ time but we conjecture that the minimum spanning tree of a greedy triangulation can be computed without computing the greedy triangulation completely. Hence, it might be possible to improve the time complexity of this step of our algorithm further.

In the second direction, we present two NP-hardness results. These results show two generalizations of the minimum weight triangulation problem to be NP-hard. For our first NP-hardness result we formulate the minimum weight triangulation as a graph theoretic problem. This provides a new way of looking at the minimum weight triangulation problem. The second NP-hardness result is for a geometric decision problem that is a generalization of the minimum weight triangulation problem. Motivated by our results in both directions we present two conjectures, which if true will determine the complexity of the minimum

weight triangulation problem completely.

Conjecture 2: The MWT problem is NP-hard.

In our graph theoretic formulation of the minimum weight triangulation problem we defined the class of graphs called crossing graphs. The determination of the computational complexity of standard graph problems such as minimum vertex cover, for the family of crossing graphs might be of interest in itself. Alternate characterizations of crossing-graphs are currently lacking but would help in designing algorithms. Solutions to these problems may provide some insight into the minimum weight triangulation problem.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *Data Structures and Algorithms.* Addison Wesley Publishing Company, 1980.

[2] M.A. Buckingham, *Circle Graphs.* Ph. D. dissertation, New York University, 1981.

[3] J.C. Davis, M. J. McCullagh, eds., *Display and Analysis of Spatial Data.* Wiley, 1975.

[4] M.R. Garey, D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness.* Freeman, San Fransisco, 1979.

[5] P.D. Gilbert, *New results on planar triangulations.* M.S. thesis, Report No. UILUENG 78 2243, University of Illinois, 1979.

[6] S.A. Goldman, *A space efficient greedy triangulation algorithm*, MIT Technical Report, Report No. MIT/LCS/TM-366, 1988.

[7] D.G. Kirkpatrick, A note on Delaunay and optimal triangulations, *Information Processing Letters* **10,** 1980, 127–128.

[8] C. Levcopoulos and A. Lingas, On approximation behavior of the greedy triangulation for convex polygons, *Algorithmica* **2,** 1987, 175–193.

[9] C. Levcopoulos, An $O(\sqrt{n})$ lower bound for the non-optimality of the greedy triangulation, *Information Processing Letters* **25,** 1987, 247–251.

[10] A. Lingas, The greedy and Delaunay triangulations are not bad in the average case and minimum weight triangulation of multi-connected polygons is NP-complete, *Foundations of Computation Theory*, Lecture Notes in Computer Science 158, Springer-Verlag, Heidelberg, 1983, 270–284.

[11] A. Lingas, The greedy and Delaunay triangulations are not bad in the average case, *Information Processing Letters* **22**, 1986, 25–31.

[12] A. Lingas, A new heuristic for the minimum weight triangulation, *SIAM Journal of Algebraic and Discrete Methods* **8**, 1987, 646–658.

[13] E.L. Lloyd, On triangulations of a set of points in the plane, *Proceedings of the Eighteenth IEEE Symposium on Foundations of Computer Science*, 1977, 228–240.

[14] G.K. Manacher and A.L. Zobrist, Neither the greedy nor the Delaunay triangulation of a planar point set approximates the optimal triangulation, *Information Processing Letters* **9**, 1979, 31–34.

[15] S. Olariu, S. Toida and M. Zubair, On a conjecture by Plaisted and Hong, *Journal of Algorithms* **9**, 1988, 597–598.

[16] D.A. Plaisted and J. Hong, A heuristic triangulation algorithm, *Journal of Algorithms* **8**, 1987, 405–437.

[17] F. Preparata and M.I. Shamos, *Computational Geometry*, Springer Verlag, 1985.

[18] M.I. Shamos and D. Hoey, Closest point problems, *Proceedings of the Sixteenth IEEE Symposium on Foundations of Computer Science*, 1975, 151–162.

[19] W.D. Smith, *Studies in Computational Geometry Motivated by Mesh Generation,* Ph.D. dissertation, Princeton University, 1989.

[20] K.J. Supowit, *Topics in Computational Geometry*, Report Number UIUCDCS-R-81-1062, UILU-ENG81 1710, University of Illinois, 1971.

[21] M. Yannakakis and F. Gavril, Edge dominating sets in graphs, *SIAM Journal of Applied Mathematics* **38**, 1980, 364–372.

[22] F.F. Yao, Efficient Dynamic Programming Using Quadrangle Inequalities. *Proceedings of the Twelfth ACM Symposium on Theory of Computing*, 1980, 429–435.

[23] F.F. Yao, Speed-up in dynamic programming, *SIAM Journal of Algebraic and Discrete Methods* **3**, 1982, 532–540.