アルゴリズム 53−13
(1996. 9. 13)

LMT-skeleton に関する一考察

加藤直樹[1], Siu-Wing Cheng[2], 菅井学[3]

[1]神戸商科大学 管理科学科, [2]Department of Computer Science, The Hong Kong University of Science & Technology. [3]神戸商科大学 大学院経営学研究科経営情報科学専攻

アブストラクト：本稿では最小重み三角形分割の部分グラフである *LMT*-skeleton を見つけるアルゴリズムについての改良を提案する。*LMT*-skeleton は Belleville et al, と Dickerson and Montague によって独立に提案されたものである。われわれの改良点はつぎの通りである。(1) *LMT*-skeleton を見つけるための Dickerson and Montague の方法における局所最小性の条件を少し緩めている。(2) Dickerson and Montague の方法を実行する際の計算量を改善している。(3) 大きなサイズの問題を解く際のボトルネックとなる記憶容量を実際的観点からかなり減らしている。これらの改良点に基づいて計算実験を行なった。

A Study of the *LMT*-skeleton

Naoki Katoh[1], Siu-Wing Cheng[2], and Manabu Sugai[1]

[1]Kobe University of Commerce
[2]Department of Computer Science, The Hong Kong University of Science & Technology

Abstract: We present improvements in finding the *LMT*-skeleton, which is a subgraph of all minimum weight triangulations, independently proposed by Belleville et al, and Dickerson and Montague. Our improvements consist of: (1) A criteria is proposed to identify edges in all minimum weight triangulations, which is a relaxation of the definition of local minimality used in Dickerson and Montague's method to find the *LMT*-skeleton; (2) A faster algorithm is presented for performing one pass of Dickerson and Montague's method (with our new criteria); (3) Improvements in the implementation that may lead to substantial space reduction for uniformly distributed point sets. In solving the minimum weight triangulation problem, finding a large subgraph is a very useful preprocessing step because a large subgraph tends to be well connected and the number of connected components appears in the exponent of the running time of an exhaustive search algorithm.

1 Introduction

Given a planar point set S, the weight of a triangulation of S is defined to be the sum of Euclidean lengths of the edges. The *minimum weight triangulation problem* is to compute a triangulation of S with minimum weight (denoted by MWT from now on). The complexity of finding a MWT is currently unresolved: it is not known to be in P or NP. Nevertheless, many new results are discovered recently concerning its geometric properties [4, 5, 7, 9, 12, 13] and combinatorial properties [2], as well as approximation algorithm [10], fast heuristics [8, 11], and algorithms for finding large subgraphs of a MWT [3, 6].

This paper is a further study on the new subgraph of a MWT independently proposed in [3, 6]. Following [6], we call this subgraph the LMT-skeleton. Although it can be proved that the LMT-skeleton can have linearly many holes in the worst case [4], the experimental results observed so far is encouraging. As indicated by the experimental results in [6], the LMT-skeleton is often connected and contains a large percentage of the edges in MWT for uniformly distributed point set (up to 250 points are tried by Dickerson and Montague [6] and up to 1000 points are tried by Belleville et al [3]). Unfortunately, finding a LMT-skeleton currently takes $O(n^4)$ time and $\Theta(n^2)$ space [3]. The implementation in [6] uses $O(n^3)$ space, runs in multiple passes, and each pass takes $O(n^4)$ time. It is not clear if a constant number of passes suffice to construct the LMT-skeleton although this is observed to be the case experimentally. The quadratic space requirement is especially a hindrance of experimenting with larger point sets.

In this paper, we pursue improved ways to find a large subgraph of a MWT based on the LMT-skeleton. First, we propose a relaxed version of the criteria used in [3, 6] for identifying edges in the LMT-skeleton. As in [6], we can apply this relaxed criteria in multiple passes to find the LMT-skeleton. Our second contribution is a more worst-case efficient algorithm for running one pass, which takes $O(n^3 \log n)$ time and $O(n^2)$ space. Space is one bottleneck for trying large point sets. The implementation in [3] uses $\Theta(n^2)$ because each point p is associated with a list of all the other points sorted in angular order around p. The implementation in [6] uses atleast $\Omega(n^2)$ space and at most $O(n^3)$ space becasue all possible edges and all empty triangles are stored. We suggest improvements in the implementation, which may lead to substantial reduction in space for uniformly distributed point sets. In our experiments, the space usage is about five times the point set size. We also ran our implementation on some data from natural resources: cross-sections of some human organs. We observe that one pass of the algorithm may generate a highly disconnected subgraph. This contrasts with the findings for uniformly distributed points.

In Section 2, we introduce our relaxed criteria and prove its correctness. Section 3 describes our improved algorithm for running one pass. In this extended abstract, due to the space limit, we omit implementation details and experimental results which will be reported in the full version.

2 A relaxed criteria

We denote the MWT of a point set S by MWT(S) and the LMT-skeleton by LMT(S). The set of all possible edges connecting two points in S is $E(S)$. Let e be an edge in $E(S)$ and t_1 and t_2 be two triangles such that $t_1 \cap t_2 = e$. In [6], e is defined to be *locally minimal* with respect to $t_1 \cup t_2$ if $t_1 \cup t_2$ is not convex, or if $t_1 \cup t_2$ is convex and $|e| \leq |e'|$, where e' is the other diagonal of $t_1 \cup t_2$. A triangle $T(S)$ of S is locally minimal if each edge e in $T(S)$ is locally minimal with respect to the two triangles containing e.

Given the above definition, it follows immediately that MWT(S) is a locally minimal triangulation and therefore, the intersection of all locally minimal triangulations must be a subgraph of MWT(S). Denote such an intersection by $\mathcal{L}(S)$. It is not clear how to compute $\mathcal{L}(S)$ without enumerating all locally minimal triangulations. But the enumeration itself is very time-consuming. A subset of $\mathcal{L}(S)$, which is the LMT(S), is computed as follows (our description is adapted from [6] and is not identical to that in [6]).

A list *candEdges* of candidate edges, a list *edgesIn* of edges known to be contained in MWT(S), and a list *deadEdges* of edges known to be not contained in any locally minimal triangulations are maintained. Initially, all edges in $E(S)$ except the convex hull edges are in *candEdges*, *edgesIn* contains the convex hull edges, and *deadEdges* is empty.

For each unexamined edge $e \in$ *candEdges*,

1. Find all combinations of empty triangles t_i and t_j on the two sides of e such that t_i and t_j are not bordered by any edge in *deadEdge*.

2. Test each combination of t_i and t_j to see if e is locally minimally with respect to t_i and t_j. If e is not locally minimal to any such pair t_i and t_j, then move e to *deadEdges*. Otherwise, if e intersects no other edge in *candEdges* or *edgesIn*, then move e to *edgesIn*.

The above is one pass to find edges in LMT(S). Multiple passes should be run until no more edges in *candEdges* can be moved to *edgesIn* or *deadEdges*. The correctness relies on the fact that an edge e is in some locally minimal triangulation *only if* e is locally minimal with respect to all neighboring empty triangles, excluding those that are neighboring to edges known to be not contained in any locally minimal triangulations. (Since such triangles cannot exist in any locally minimal triangulations, e needs not be locally minimal with respect to them.) Therefore, the above algorithm works by first eliminating all edges that cannot be contained in any locally minimal triangulations. Such edges are put in *deadEdges* and they are called *dead edges*. Each remaining edge is possibly contained in some locally minimal triangulation and we call them *active edges*. Thus, if an active edge does not intersect any other active edge, then it belongs to $\mathcal{L}(S)$ and can be included in LMT(S) (i.e., put in *edgesIn*). It is not clear that $LMT(S) = \mathcal{L}(S)$ because it is possible but not certain that an active edge is in some locally minimal triangulation.

In the previous definition, if there are neighboring triangles t_1 and t_2 bordering e such that $t_1 \cup t_2$ is non-convex, then e is locally minimal with respect to t_1 and t_2 and so e will become one of the remaining edges. However, the fact that $t_1 \cup t_2$ is non-convex implies that $t_1 \cup t_2$ does not really play any active role. Thus, we perhaps should not label e as an active edge immediately based on t_1 and t_2. For example, if e is not locally minimal to any other pairs of empty triangles, then we may not want to label e as active. After all, excluding e means that it is more likely to include more active edges in LMT(S). This prompts us to define a weaker criteria in the following.

First, we introduce some terminology to ease our discussion. All edges in $E(S)$ and all empty triangles are initially active. Edges will become *dead* if they are known to be not contained in any locally minimal triangulation. Whenever an edge ab becomes dead, we collect the set $\mathcal{A}$ of all empty active triangles axb satisfying: for all empty active triangles ayb such that $axb \cap ayb = ab$, $ab \cap xy \neq \emptyset$. Then we label all triangles in $\mathcal{A}$ dead. When each edge is examined, its status will be determined as active, *inactive*, or dead as follows:

> Let $\mathcal{T}$ be the set of pairs $\{axb, ayb\}$ of empty active triangles such that $axb \cap ayb = ab$. An edge ab is labelled active if it lies on the boundary of the convex hull, or there exists $\{axb, ayb\} \in \mathcal{T}$ such that $ab \cap xy \neq \emptyset$ and $|ab| \leq |xy|$. Suppose that ab is not labelled active. Then if $\mathcal{T} = \emptyset$ or $ab \cap xy \neq \emptyset$ for all $\{axb, ayb\} \in \mathcal{T}$, then we label ab dead. Otherwise, we label ab inactive.

Clearly, our criteria is no stronger than the definition of locally minimal edges. That is, if an edge e is labelled active by our criteria, then e must also be labelled active by the definition of locally minimality. After finding the active edges, an active or inactive edge is in LMT(S) if it does not intersect any other active edge. We prove the correctness of our criteria in the following.

Lemma 1 *If an empty triangle t is labelled dead, then $t \notin MWT$(S).*

PROOF. We prove the lemma by contradiction. Let axb be the first triangle in MWT(S) which is labelled dead and suppose that it is labelled dead because the edge ab becomes dead. So ab does not lie on the boundary of $conv(S)$ and so ab is adjacent to some triangle ayb in MWT(S) such that $y \neq x$. Since axb is the first triangle becoming dead, ayb is active when ab is to be examined. Moreover, the fact that axb becomes dead implies that $ab \cap xy \neq \emptyset$. However, the fact that ab becomes dead implies that $|xy| < |ab|$. Therefore, we can replace ab by xy to decrease the total length of MWT(S), a contradiction. □

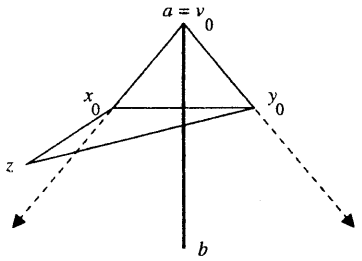


Fig. 1.

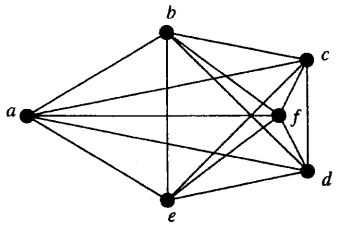


Fig. 2.

Lemma 2 *If $ab \notin MWT(S)$, then ab intersects some active edge.*

PROOF. Assume to the contrary that $ab \notin MWT(S)$ but ab does not intersect any active edge. Since $ab \notin MWT(S)$, ab intersects some edge in $MWT(S)$. Let x_0 and y_0 be the edge in $MWT(S)$ such that $ab \cap x_0y_0$ is closest to a. Thus, $x_0ay_0 \in MWT(S)$. For convenience, we also rename a as v_0. Let C_0 be the cone bounded by the two rays originating from v_0 through x_0 and y_0, respectively. By construction, $b \in C_0$. Consider the other triangle x_0zy_0 adjacent to x_0y_0 in $MWT(S)$. See Figure 1.

By Lemma 1, $x_0v_0y_0$ and x_0zy_0 were active when x_0y_0 was labelled dead in **test**(x_0y_0). So we conclude that $v_0z \cap x_0y_0 = \emptyset$, otherwise, $|v_0z| < |x_0y_0|$ and we can flip x_0y_0 to decrease the total length of $MWT(S)$, which is impossible. Therefore, $z \notin C_0$ and $z \neq b$. Without loss of generality, let z lies on the left of v_0b. We rename x_0 as v_1, z as x_1, and y_0 as y_1. By construction, we discover a new edge x_1y_1 in $MWT(S)$ that intersects v_0b, and b lies inside the cone C_1 bounded by the two rays from v_1 through x_1 and y_1. Thus, we can repeat the previous argument again to $x_1v_1y_1$ and x_1y_1 to obtain a new edge x_2y_2 that intersects v_0b. In fact, we can repeat this argument indefinitely to obtain an infinite sequence of edges x_iy_i, $i > 1$, that intersect v_0b. This contradicts the finiteness of $MWT(S)$. □

Lemma 2 implies the correctness of reporting active or inactive edges that do not intersect any other active edges. We have mentioned before that our criteria is no stronger than the definition of locally minimal edges. There is a point set such that in one pass, a larger subgraph of $MWT(S)$ will be reported if our criteria is used instead of the definition of locally minimal edges. Consider the point set X in Figure 2. $MWT(X)$ contains the convex hull edges and be, bf, ef, cf, and df. Other the convex hull edges, suppose that $E(S)$ is processed in this order af,ac, ad, ce, bd, be, bf, ef, cf, df. Using the definition of locally minimal edges, ac, ad, ce, and bd will be labelled dead; af, cf, df, bf, and ef will be labelled active. Among the active edges, cf, df, and ef will be reported to be in $LMT(S)$. The edge be is not reported due to the presence of af. In contrast, using our criteria, af will be labelled inactive and so cf, df, ef as well as bf will be reported to be in $LMT(S)$. (In fact, cf and df will also be labelled as inactive.)

3 An efficient algorithm

We present an algorithm that runs in $O(n^3 \log n)$ time. The worst-case space usage is $O(n^2)$. The improvement comes from two techniques: (1) when we examine an edge e, instead of enumerating every pair of empty active triangles bordering e, we process all of them simultaneously by looking at two simple polygons induced by these triangles, (2) for each active or inactive edge e, we develop a more efficient way to check if e intersects an active edge.

To ease the discussion, for the time being, whenever an edge becomes dead, we will not label any empty active triangle bordering the edge dead. We will discuss how to put this feature back later. Given a direction α, we use $S_p(\alpha)$ to denote the sequence of points in $S - \{p\}$ swept by

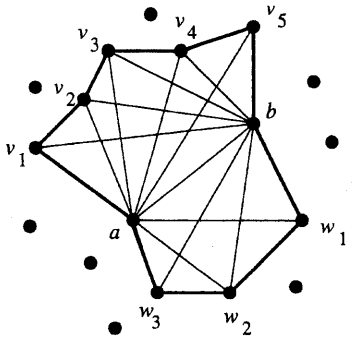


Figure 4: The bold polygonal figure is $L_{ab} \cup R_{ab}$. Note that the triangles being empty does not imply that $L_{ab} \cup R_{ab}$ is empty.

rotating a ray around p in counterclockwise order starting from direction α. Given any point $x \in S - \{p\}$, we use $S_p(\alpha, x)$ to denote the subset of points that precede x in $S_\alpha(p)$. The following simple lemma enables us to identify the empty triangles adjacent to an edge ab.

Lemma 3 *Let α be the direction denoted by the ray from point a to point b. Given a triangle axb lying on the left of ab with respect to α, axb is empty iff $S_a(\alpha, x) \subseteq S_b(\alpha, x)$.*

PROOF. If y lies outside axb and $y \in S_a(\alpha, x)$, then y must lie inside the halfspace defined by the supporting line of bx that does not contain a. This implies that $y \in S_b(\alpha, x)$ and proves the forward direction. If y lies inside axb, then $y \in S_a(\alpha, x)$ but $y \notin S_b(\alpha, x)$. This proves the reverse direction. □

Let α be the direction of a to b. By Lemma 3, we can find all the points x such that axb is empty and lies on the left of ab with respect to α in $O(n \log n)$ time as follows. Sort points around a and b separately. Scan $S_a(\alpha)$ and for each point x, visit the corresponding entry in $S_b(\alpha)$. If the rank of x in $S_b(\alpha)$ is found to be larger than the ranks of all previously visited entries in $S_b(\alpha)$, then axb is an empty triangle. We connect the set $\{x : axb \text{ is an empty triangle}\}$ in counter-clockwise order to form a simple polygon L_{ab}. Similarly, we can construct a simple polygon R_{ab} on the right of ab with respect to α. Refer to Figure 4 for an illustration. Now, it suffices to find the longest diagonal e of $L_{ab} \cup R_{ab}$ that intersects ab. If e exists and $|e| \geq |ab|$, then we label ab active. Suppose that ab is not labelled active. If L_{ab} or R_{ab} is empty, or if the angle at a or b in $L_{ab} \cup R_{ab}$ is not larger than π, then we label ab dead. Otherwise, we label ab inactive. The labelling of ab can be done in quadratic time by brute-force. In fact, this is the preferred method whenever the size of $L_{ab} \cup R_{ab}$ is not greater than $\sqrt{n}$ because this incurs less overhead. (This case will apply most of the time if there are not too many empty triangles adjacent to ab.) We need a different method if the size of $L_{ab} \cup R_{ab}$ is greater than $\sqrt{n}$.

Name the vertices of L_{ab} on the chain from a to b (excluding a and b) as $v_1, v_2, \cdots, v_l$. Name the vertices of R_{ab} on the chain from b to a (excluding a and b) as $w_1, w_2, \cdots, w_m$. First, observe that each w_i can see a contiguous subsequence of vertices $\{v_{f(i)}, v_{f(i)+1}, \cdots, v_{g(i)}\}$ on L_{ab}. Moreover, the sequences $\{f(i) : 1 \leq i \leq m\}$ and $\{g(i) : 1 \leq i \leq m\}$ are non-decreasing. Thus, the "intervals" of visible vertices from all w_i's can be computed by a single traversal of the chain on L_{ab} from b to a. Hence, to find the longest diagonal of $L_{ab} \cup R_{ab}$ that intersects ab, it suffices to find efficiently the furthest vertex from w_i among those on L_{ab} visible from w_i.

Lemma 4 *Given w_p and w_q with $p < q$ and v_r and v_s with $r < s$, if v_r is visible from w_q and v_s is visible from w_p, then v_r is visible from w_p and v_s is visible from w_q. Moreover, for $1 \leq i \leq m$ and $1 \leq j \leq l$, define $d_{ij} = |w_i v_j|$ if v_j is visible from w_i or $-\infty$ otherwise. Then (d_{ij}) is a Monge matrix.*

PROOF. Since v_r is visible from w_q and v_s is visible from w_p, $aw_ib \cup av_jb$ is convex for all w_i between w_p and w_q and v_j between v_r and v_s. Thus, we conclude that v_r is visible from w_p and v_s is visible from w_q. Take any four entries $d_{i_1j_1}, d_{i_1j_2}, d_{i_2j_1}, d_{i_2j_2}$ of the matrix (d_{ij}), where $i_1 > i_2$ and $j_1 > j_2$. If v_{j_2} is not visible from w_{i_1} or v_{j_1} is not visible from w_{i_2}, then $d_{i_1j_2} = -\infty$ or $d_{i_2j_1} = -\infty$. So $d_{i_1j_1} + d_{i_2j_2} \geq d_{i_1j_2} + d_{i_2j_1}$. If v_{j_2} is visible from w_{i_1} and v_{j_1} is visible from w_{i_2}, then we know that v_{j_1} is visible from w_{i_1} and v_{j_2} is visible from w_{i_2}. Given any convex quadrilateral, the sum of the lengths of the two diagonals is always no less than the sum of lengths of any two opposite sides. Thus, $d_{i_1j_1} + d_{i_2j_2} \geq d_{i_1j_2} + d_{i_2j_1}$. This proves that (d_{ij}) is a Monge matrix. □

We can then compute in $O(n)$ time the furthest visible vertex on L_{ab} from w_i for each i by applying fast matrix searching algorithm on a Monge matrix [1]. In all, we can determine if ab is active, inactive, or dead in $O(n^3 \log n)$ time.

To handle labelling empty triangles dead, for each edge e whose status has been determined, we maintain a *window* of e, *window*(e), which consists of the four edges of $L_e \cup R_e$ adjacent to e. Let ab be the edge whose status is to be determined. Suppose that during the scanning of $S_a(\alpha)$ and $S_b(\alpha)$ to produce L_{ab} or R_{ab}, we encounter a point x in $S_a(\alpha)$ such that axb is detected to be empty but ax is dead. (The symmetric case for bx is handled similarly.) There are two edges $e_1 = ac$ and $e_2 = dx$ in *window*(ax) that lie on the side of ax opposite to that containing b. If $bc \cap ax \neq \emptyset$ and $bd \cap ax \neq \emptyset$, then axb should be labelled dead and we do not include x in forming the simple polygon. After L_{ab} and R_{ab} are obained, we process them as described before.

In all, we determine the status of all edges in $E(S)$ in $O(n^3 \log n)$ time if we apply one pass. The space needed is $O(n^2)$. The next task is to report all the active and inactive edges that do not intersect an other active edge.

We shall examine each point $p \in S$ and determine for each active or inactive edge e incident to p, whether e intersects any active not incident to p. For each point $p \in S$, we first compute the "envelope" of active edges not incident to p around p. That is, imagine that we emit light rays from p in all directions, we wan to find out all the points on active edges (not incident to p) hit by these light rays. The situation is basically the same as computing the lower envelope of a set of line segments with at most n distinct endpoints. Recall that there are possibly $O(n^2)$ active edges. We describe below an $O(n^2 \log n)$ divide-and-conquer algorithm to compute the lower envelope of $O(n^2)$ line segments with at most n endpoints. This algorithm can be adapted to compute the "envelope" around p.

Given $O(n^2)$ line segments, we first sort their endpoints by x-coordinates in $O(n^2 + n \log n)$ time. Then we split the set of endpoints into two equal halves by a vertical splitting line ℓ. We take out all the line segments that intersecting ℓ. All remaining line segments lie completely on one side of ℓ. We first recursively find out the lower envelope of line segments on both sides of ℓ. Then we compute the lower envelope of line segments intersecting ℓ and then merge the three envelopes to obtain the final solution. The complexity of a lower envelope of $O(n^2)$ line segments is $O(n^2\alpha(n,n))$, where $\alpha(n,n)$ is the inverse Ackermann function. Therefore, if the lower envelope of line segments intersecting ℓ can be found in $O(n^2 \log n)$ time, then the merging step can be done in $O(n^2 \log n)$. Thus, we have the following recurrence equation $T(n) \leq 2T(n/2) + O(n^2 \log n)$ and it solution is $O(n^2 \log n)$. The lower envelope of line segments intersecting ℓ can be found as follows. It suffices to focus on the left half of this lower envelope on the left of ℓ as the other half can be handled similarly. We sort the line segments into the order $s_1, s_2, \cdots$, such that the left endpoint of s_i is further away from ℓ than the left endpoint of s_{i+1}. Our approach is to insert segments $s_1, s_2, s_3, \cdots$ in this order to construct the solution. Suppose that we have already found the left half of the lower envelope for $s_1, s_2, \cdots, s_{i-1}$. In general, it consists of a number disjoint concave chains ordered from left to right. By the ordering, s_i can only intersect the right most concave chain C in this intermediate solution. If we represent C using a balanced search tree, then we can determine the intersection between s_i and C and find the lower envelope of C and s_i in $O(\log n)$. (Note this lower envelope may consists of two disjoint concave chains.) This together with the other concave chains form the left half of the lower envelope of $s_1, \cdots, s_i$. Since there are

Table 1: For each size from 100 to 500, 50 random trials were performed. For size 1000, 10 random trials were performed. For size 2000, 5 random trials were performed.

Size	Aver. # comp.	Aver. # active/inactive edges	Aver. % MWT
100	1.04	406.14	77.15
200	1.36	841.84	75.73
300	1.24	1298.96	74.72
400	1.62	1781.46	73.62
500	1.34	2239.70	74.04
1000	1.6	4630.9	73.38
2000	2.8	9422	73.88

$O(n^2)$ line segments intersecting ℓ, the total time needed is $O(n^2 \log n)$.

After computing the "envelope" of active edges around p which has $O(n^2\alpha(n,n))$ complexity, we connect each vertex of this "envelope" with p to form a set of wedges around p. Now, for each active or inactive edge e incident to p, we can perform a binary search to find the wedge containing e. Then e intersects an active edge not incident to p if and only if e intersects the side of the wedge not incident to p. In all, it takes $O(n^2 \log n)$ time to process on point in S and it sums to a total time of $O(n^3 \log n)$.

4 Experiments

We have run *one pass* of our implementation on uniformly point sets of size 100, 200, 300, 400, 500, 1000, and 2000. Figure 5(a)–(e) in the appendix show the MWT edges identified for example point sets of size 100, 200, 300, 400, and 500. Table 1 shows the average number of connected components in the resulting graph, the average number of active and inactive edges found, and the average percentage of MWT edges found. For all the point sets, we identify at least 73% of the MWT edges on average. For point set of sizes from 100 to 1000, we almost always obtain one single connected component. For point sets of size 2000, it is more often to obtain more than one connected component in one pass (the number of connected component goes up to four in some test case with 2000 points). We would like to emphasize again that this is done *in just one pass.* In all our experiments with uniformly distributed point sets, we observe that the number of active and inactive edges identified is always bounded by five times the size of the point set. This helps to explain the roughly cubic running time observed and this also suggests that our implementation may use significantly less than quadratic space for uniformly distributed point sets.

Other than uniformly distributed point sets, we have also run one pass of our implementation on a few two-dimensional cross-sections of some human organs. But the details of our results are omitted here.

For comparison, we also run one pass of Dickerson and Montague's algorithm on edges in decreasing lengths for uniformly distributed point sets of size 500 and 1000. Our implementation performs slightly better in identifying more MWT edges. Thus, in comparison, the major advantage of our implementation is the saving in space which permits us to experiment with larger point sets.

References

[1] A. Aggarwal, M.M. Klawe, S. Moran, P.W. Shor, and R. Wilber: *Geometric applications of a matrix-searching algorithm*, Algorithmica, 2 (1987), pp. 195–208.

[2] O. Aichholzer, F. Aurenhammer, S.W. Cheng, N. Katoh, M. Taschwer, G. Rote, and Y.F. Xu, *Triangulations intersect nicely*, manuscript, 1996.

[3] P. Belleville, M. Keil, M. McAllister, and J. Snoeyink, *On computing edges that are in all minimum-weight triangulations*, Video Presentation, Symp. Computational Geometry, 1996, to appear.

[4] P. Bose, L. Devroye, and W. Evans, *Diamonds are not a minimum weight triangulation's best friend*, Technical Report 96-01, Dept. of Computer Science, Univ. of British Columbia, January 1996.

[5] S.W. Cheng and Y.F. Xu, *Approaching the largest β-skeleton within a minimum weight triangulation*, Proc. Symp. Computational Geometry, 1996, to appear.

[6] M.T. Dickerson and M.H. Montague, *The Exact Minimum Weight Triangulation*, Proc. Symp. on Computational Geometry, 1996, to appear.

[7] M. Golin, *Limit theorems for minimum-weight triangulations, other Euclidean functionals, and probabilistic recurrence relations*, Proc. Symp. Discrete Algorithms, 1996, pp. 252–260.

[8] L. Heath and S.V Pemmaraju, *New results for the minimum weight triangulation problem*, Algorithmica, 12 (1994), pp. 533–552.

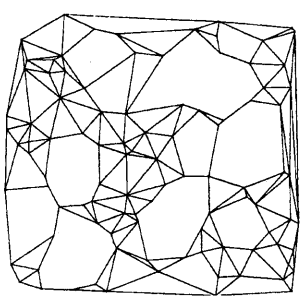
[9] J.M. Keil, *Computing a subgraph of the minimum weight triangulation*, Computational Geometry: Theory and Applications, 4 (1994), pp. 13–26.

[10] C. Levcopoulos and D. Krznaric, *Quasi-greedy triangulations approximating the minimum weight triangulation*, Proc. Symp. Discrete Algorithms, 1996, pp. 392–401.

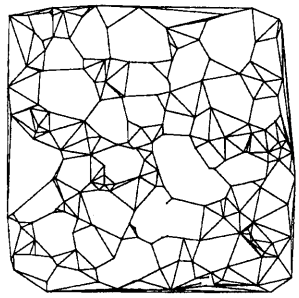
[11] C. Levcopoulos and D. Krznaric, *A fast heuristic for approximating the minimum weight triangulation*, Proc. Scandinavian Workshop on Algorithmic Theory, 1996, to appear.

[12] B. Yang, *A better subgraph of the minimum weight triangulation*, in Proc. International Conference on Computing and Combinatorics, 1995, pp. 452–455.

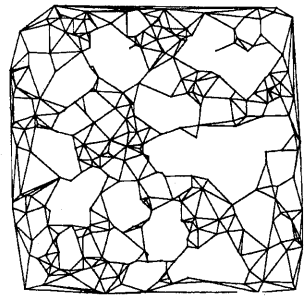
[13] B. Yang, Y. Xu, and Z. You, *A chain decomposition algorithm for the proof of a property on minimum weight triangulation*, in Proc. International Symposium on Algorithms and Computation, 1994, pp. 423–427.



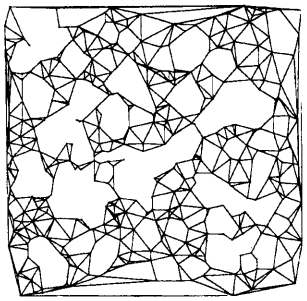
(a) 100 points



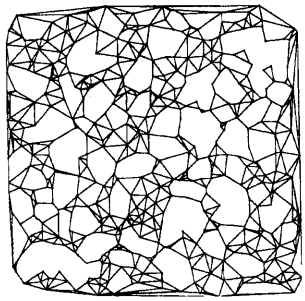
(b) 200 points



(c) 300 points



(d) 400 points



(e) 500 points

Figure 5: