Discrete Comput Geom 18:289–304 (1997)

Discrete & Computational
Geometry

A Large Subgraph of the Minimum Weight Triangulation*

M. T. Dickerson,[1] J. M. Keil,[2] and M. H. Montague[1]

[1]Department of Mathematics and Computer Science, Middlebury College,
Middlebury, VT 05753-6145, USA
dickerso@middlebury.edu

[2]Department of Computer Science, University of Saskatchewan,
Saskatoon, Saskatchewan, Canada
keil@cs.usask.ca

Abstract. We present an $O(n^4)$-time and $O(n^2)$-space algorithm that computes a subgraph of the *minimum weight triangulation* (*MWT*) of a general point set. The algorithm works by finding a collection of edges guaranteed to be in any *locally minimal* triangulation. We call this subgraph the *LMT*-skeleton. We also give a variant called the *modified LMT*-skeleton that is both a more complete subgraph of the *MWT* and is faster to compute requiring only $O(n^2)$ time and $O(n)$ space in the expected case for uniform distributions. Several experimental implementations of both approaches have shown that for moderate-sized point sets (up to 350 points[1]) the skeletons are connected, enabling an efficient completion of the exact *MWT*. We are thus able to compute the *MWT* of substantially larger random point sets than have previously been computed.

1. Introduction

Given a planar point set S and a triangulation $T(S)$, the weight of $T(S)$ is the sum over all edges e in $T(S)$ of the weight of e (denoted $wt(e)$). The weight of an edge is defined here as its Euclidean length.[2] A *minimum weight triangulation* (*MWT*) of a point set

* First presented at the Workshop on Computational Geometry (WOCG) in Stonybrook, NY in October 1995, with a preliminary version also appearing in [10]. The research of the first and third authors was supported in part by the funds of the National Science Foundation, NSF CCR-9301714. The second author's study was supported by NSERC under Grant OGP0000259. M. H. Montague is now at Dartmouth College, Hanover, NH, USA.

[1] Though in this paper we summarize some empirical findings for input sets of up to 350 points, a variant of the algorithm has been implemented and tested on up to 40,000 points producing connected subgraphs [2].

[2] There are alternate possible definitions of weight. We note that our underlying *algorithm* is not dependent on any particular definition, but our experimental results clearly follow from defining weight as Euclidean length.

S is a triangulation that minimizes weight over all possible triangulations. The *MWT* problem is to find the *MWT* of a given input set S.

Though it has been shown how to compute $MWT(P)$ in time $O(n^3)$ when P is a simple polygon [14], [16], the complexity status of the *MWT* problem for a general point set is unknown [13]. It has been a longstanding open problem to give an efficient *MWT* algorithm or to prove the problem NP-hard.

1.1. *Overview*

In this paper, we present an $O(n^4)$-time and $O(n^2)$-space algorithm that computes a subgraph of the *MWT* of a general point set. The algorithm works by finding a collection of edges guaranteed to be in any *locally minimal* triangulation. We call this subgraph the *LMT*-skeleton. We also give a faster variant of our algorithm that produces a more complete subgraph of the *MWT* in $O(n^2)$ time and $O(n)$ space in the expected case for uniform distributions. We call this second approach the *modified LMT*-skeleton. Several experimental implementations of both approaches have shown that for moderate-sized point sets (up to 350 points) the skeletons are connected, enabling an efficient completion of the exact *MWT*. We are thus able to compute the *MWT* of substantially larger random point sets than have previously been computed.

The theory behind these skeletons is not complex. What is perhaps surprising is how well they have worked in practice. Our main purpose in this paper is to present the *LMT*-skeleton, the variant of the skeleton called the *modified LMT*-skeleton, and the algorithms to produce them. We also report on some highly promising experimental data from our implementations. Empirical results indicate that our *LMT*-skeleton contains many more edges than any previously known efficiently computable subgraphs such as those of [5], [15], and [19]. In fact, in over 2000 trials on uniform random distributions of up to 350 points, the algorithm has produced a connected subgraph in every case. From the *LMT*-skeleton subgraphs, we are therefore able to produce a complete exact *MWT*, such as that shown in Fig. 1.

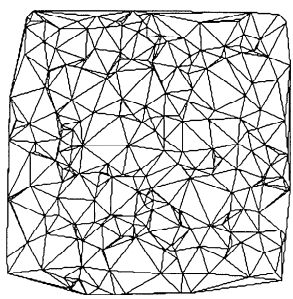

Fig. 1. An exact *MWT* of 250 points. The figure shows the *MWT* of a set of 250 points taken in a uniform random distribution of over a square. No Steiner points were added to make the triangulation easier. The triangulation was computed from the *LMT*-skeleton.

1.2. *Previously Known Subgraphs of the MWT*

As noted, we can compute $MWT(P)$ of a simple polygon P in $O(n^3)$ time. Thus if we can compute a *connected* subgraph of $MWT(S)$, then after adding the convex hull we can view the remaining untriangulated regions as simple polygons, and can then complete $MWT(S)$ in polynomial time. Since there is no known efficient algorithm to solve the exact *MWT* problem in the general case, recent investigations have generally concentrated either on finding good approximations of the $MWT(S)$ (see, for example, [10]) or on finding an extensive *subgraph* of $MWT(S)$.

The convex hull, being a subgraph of any triangulation, is (trivially) a subgraph of $MWT(S)$. Gilbert [14] showed that the shortest possible edge in S is also in $MWT(S)$. More recently and less trivially, Yang *et al*. [19] showed that all mutual nearest-neighbor edges are in $MWT(S)$. Keil [15] showed that the so-called "β-skeleton" of S is a subgraph of $MWT(S)$ when $\beta = \sqrt{2}$, and Cheng and Xu [5] have recently tightened Keil's result, showing that the β-skeleton is a subgraph of $MWT(S)$ when $\beta = 1.17682$.

It is also known that the β-skeleton is not always a subgraph of the *MWT* for $\beta < 1/\sin(\pi/3)$ (or approximately $\beta < 1.1547$). In fact, there exists a four-point counterexample [15]. So there is little room for improvement on the value of β given by Cheng and Xu.

In a related work, Aichholzer *et al*. [1] discuss the concept of a *light edge* and a *light triangulation*. An edge is called *light* if there is no shorter edge crossing it. The set of all light edges forms a graph, which is shown to be a subgraph of the greedy triangulation. If the collection of all light edges forms a complete triangulation, then we call this a *light triangulation*. In [1] it is shown that if a planar point set admits a light triangulation, then the light triangulation is both the greedy triangulation and the *MWT*. They use this information to produce a point set of size 150 for which they give an exact *MWT*. Unfortunately, in general the set of light edges is not a subgraph of the *MWT*, and thus this method will not work to produce the *MWT* of an arbitrary point set.

1.3. *Locally Minimal Triangulations*

Our new result is based on the idea of *local* minimality. In any triangulation, each edge e that does not lie on the convex hull borders two empty triangles. Together these two triangles form an empty quadrilateral q, of which e is a diagonal. If q is convex, then it also has another diagonal e'. We say that e is *locally minimal* if either q is not convex (in which case q has only one triangulation—the one containing e), or if q is convex and $wt(e) \leq wt(e')$. To replace e with e' (when q is convex) is called *flipping* the edge e. A triangulation is called a *locally minimal triangulation* (*LMT*) if each of its edges is locally minimal (no edge can be flipped to produce a triangulation of lower weight).

The triangulation $MWT(S)$ must be locally minimal; otherwise some nonminimal edge could be flipped to reduce the weight of the *MWT* which is a contradiction of the definition. Unfortunately, S may allow many different locally minimal triangulations most of which are not globally minimal (not $MWT(S)$).

2. A New Subgraph of the *MWT*

In this paper we present an algorithm to compute a new subgraph of the *MWT*, which we call the *LMT*-skeleton. (We formally define the *LMT*-skeleton as any skeleton produced by Algorithm 1 in Fig. 2.) This subgraph contains a set of edges that must be in *every* locally minimal triangulation.[3] The algorithm requires $O(n^4)$ time and $O(n^2)$ space in the worst case. When the algorithm terminates, the *LMT*-skeleton can then be used to complete the *MWT* in time $O(n^3)$ if the *LMT*-skeleton is connected, or time $O(n^{k+2})$ if it has k unconnected components [6].

We also present a *modified LMT*-skeleton. The *modified LMT*-skeleton is also a subgraph of the *MWT*. In the expected case for uniformly distributed point sets, it can be computed much faster than the previous *LMT*-skeleton. We describe how to compute the *modified LMT*-skeleton in $O(n^2)$ time and $O(n)$ space in the expected case for uniform distributions. Interestingly enough, not only is the *modified LMT*-skeleton algorithm faster, but it produces more edges and is in fact a superset of the *LMT*-skeleton. However, though the *modified LMT*-skeleton is a subgraph of the *MWT*, all edges in it are not necessarily in *every* locally minimal triangulation. That is, there exist *locally* minimal triangulations not containing the entire *modified LMT*-skeleton algorithm, even though the edges do appear in every (globally) *MWT*.

Experimental results for uniform random point sets up to 350 points (the largest point sets we have tried so far) have found the *LMT*-skeleton to be connected in every case. Empirical data also suggest that the algorithm's actual average complexity is much better than $O(n^4)$ in practice. In fact, the algorithm appears to run in time closer to $O(n^3)$. Furthermore, the internal nontriangulated polygonal regions have had a small number of points (usually fewer than 15 for n up to 350), and so in practice the *MWT* has been completed from the *LMT*-skeleton in $O(n)$ additional time.

2.1. *Notation*

In the following discussion we refer to empty edges, empty triangles, and empty quadrilaterals. An *empty edge* is one that intersects no points in S other than its endpoints. An *empty triangle* is a triangle formed by three empty edges whose interior is also empty of points from S. An *empty quadrilateral* is any quadrilateral formed by two adjacent empty triangles (triangles that share a single common edge but do not intersect in their interiors).

We refer to the set of all possible edges in S as $E(S)$.

2.2. *Background and Lemmas*

Our *LMT*-skeleton subgraph of the *MWT* and our *LMT*-skeleton algorithm are based on the following observations and lemmas.

[3] Every edge in the *LMT*-skeleton is in every locally minimal triangulation, but it is not proven that every edge in every locally minimal triangulation is in our subgraph.

Observation 1. *For any planar point set S, an edge in every locally minimal triangulation of S is in $MWT(S)$, and an edge in no locally minimal triangulation of S is not in $MWT(S)$.*

The above observation (which follows directly from the local minimality of the *MWT*) describes a partition of edges into three disjoint sets: those in *all* locally minimal triangulations, those in *no* locally minimal triangulations, and the remaining edges which are in *some* (but not all) locally minimal triangulations.

The following two lemmas give properties that are easy to verify, and allow us to construct an algorithm to find some edges that must be in every locally minimal triangulation.

Lemma 1. *Let $e \in E(S)$ be an edge in at least one locally minimal triangulation. Then there exist two triangles t_1 and t_2 whose vertices are in S and which contain no other points in S, and such that t_1 and t_2 form a locally minimal quadrilateral with e as the diagonal.*

Proof. Pick any locally minimal triangulation T containing e, and let t_1 and t_2 be the triangles bounding edge e in T. □

Lemma 2. *Let e be an edge in $E(S)$. If there is no other edge in $E(S)$ that intersects e and is itself in a locally minimal triangulation, then e is in all locally minimal triangulations.*

Proof. Assume there is in some locally minimal triangulation T not containing edge e. Then T contains some edge e' crossing e which is a contradiction. □

2.3. *LMT-Skeleton Algorithm*

We can now present our algorithm[4] for computing the *LMT*-skeleton. We keep track of two lists: a list *candEdges* of candidate edges (edges of unknown status), and a list *edgesIn* of edges that have been determined to be in any locally minimal triangulation. Remaining edges are not in any locally minimal triangulation. It is also convenient to refer to a list *candTris* of candidate triangles, though for worst-case space efficiency these should not be explicitly stored but will be computed as needed. Initially, all empty triangles are candidate triangles, and all empty edges are candidate edges. Our algorithm proceeds in two ways. From the current set of candidate edges and triangles, we use Lemma 1 to eliminate any edges that we can determine are not in any $LMT(S)$. We then use Lemma 2 to find edges that are in every $LMT(S)$ (and thus in $MWT(S)$). We add these later edges to *edgesIn*. These two steps alternate iteratively. When we are done and no more edges change status, then the set *edgesIn* gives us the *LMT*-skeleton.

[4] Those familiar with the earlier conference version [11] of this paper will notice a change in our terminology; in this paper we call the *LMT*-skeleton that which we formerly referred to as the *extended-LMT*-skeleton.

1. *candEdges* := A list of all empty edges in S.
2. *edgesIn* := A new empty list.
3. Remove convex hull edges from the *candEdges* list and add them to the *edgesIn* list.
4. **REPEAT**
   (a) **FOR** each edge $e \in$ *candEdges* **DO**:
   IF there does not exist a pair of triangles $t_i \in T_L(e)$ and $t_j \in T_R(e)$ such that e is locally minimal with respect to t_i and t_j and t_i and t_j are both in *candTris*, **THEN** remove e from *candEdges* and remove all triangles containing e from *candTris*.
   END FOR

   (b) **FOR** each edge $e \in$ *candEdges* **DO**:
   IF e intersects no other edge in *candEdges*, **THEN** add e to the *edgesIn*.
   END FOR

   UNTIL no more edges change status

Fig. 2. Algorithm 1: *LMT*-skeleton.

The algorithm is given in Fig. 2. Note that in the inner **FOR** loop at Step 4a, every time an edge is eliminated from the *candEdges* list, some triangles are also eliminated from *candTris*. This might provide enough information to add later to *edgesIn* (or delete from *candEdges*) an edge which was previously examined and whose status remained unknown. This leads to the idea of the outer **REPEAT-UNTIL** loop, which repeats both **FOR** loops multiple times to produce extra edges until no more edges can be removed from *candEdges* or added to *edgesIn*.

2.4. *Correctness*

When the algorithm terminates, the edges in *EdgesIn* form a subgraph of *MWT*(S); we call this subgraph an *LMT*-skeleton. Once again, because the *LMT*-skeleton is actually *defined* by Algorithm 1, there is no proof required for the correctness of the algorithm. What we want to prove instead is simply the following:

Lemma 3. *The LMT-skeleton produced by Algorithm* 1 *is a subgraph of the MWT*.

We prove that every edge in the *LMT*-skeleton appears in every locally minimal triangulation and thus in every *MWT*. The proof of this follows almost directly from Lemmas 1 and 2. Note that our algorithm adds edges to the *LMT*-skeleton in only two places: Steps 3 and 4b. In Step 3 we add only the convex hull edges, but these must be in any triangulation. In Step 4a we remove only those edges for which there do not exist two adjacent triangles forming a locally minimal quadrilateral. Thus by Lemma 1 we know that Step 4a does not remove from *candEdges* any edges that might be in the *MWT*. In Step 4b we add only edges that are in *candEdges* but which do not intersect any other edges in *candEdges*. Since *candEdges* holds all edges that might be in some *MWT*—except those which no longer cross any others—by Lemma 2 Step 4b does not add any edges that might not be in the *MWT*.

2.5. *Complexity Analysis*

In the first step, $O(n^2 \log n)$ time and $O(n^2)$ space are required to generate the initial list *candEdges*, and to sort the list of edges around each vertex. We can compute the convex hull for Step 3 in worst-case $O(n \log n)$ time using any of a number of methods.

The most expensive step of Algorithm 1 (asymptotically) is Step 4. Potentially, the algorithm tests all empty quadrilaterals—of which there may be $\Theta(n^4)$—to see if a particular edge is locally minimal with respect to at least one empty quadrilateral. The goal is to spend $O(1)$ (amortized) time per quadrilateral, which is equivalent to spending $O(n^2)$ time per edge, in order to achieve a total running time of $O(n^4)$ in the worst case. Specifically, the inner **FOR** loops in Step 4 iterate $O(n^2)$ times, once for each candidate edge e. In Step 4a we test all quadrilaterals (pairs of adjacent triangles) in which e might be a diagonal. Since e might have a linear number of triangles on each side, we may test $O(n^2)$ quadrilaterals per edge. The convexity test requires $O(1)$ time. If the quadrilateral is convex, then we also compare two diagonal lengths, also in $O(1)$ time. In Step 4b we compare an edge against at most $O(n^2)$ other edges.

The question remains how to generate the empty triangles efficiently. If the list *candTris* is preprocessed (explicitly computed and stored in an appropriate data structure), then $O(1)$ time complexity is immediate, but the cost is $O(n^3)$ space in the worst case. If the list *candTris* is not preprocessed, then we can use a brute-force approach to compute the $O(n)$ empty triangles adjacent to edge $e = (p, q)$ in $O(n^2)$ time. For each $r \in S$ with $r \neq p, q$, test first to see if r is adjacent to both p and q in *candEdges*. This requires $O(\log n)$ time if *candEdges* are stored in a rotational-order binary search tree around each point, or $O(n)$ time in a simple unordered adjacency list. If r is adjacent to both p and q, then test the triangle pqr for emptiness in worst-case $O(n)$ time, or using bucketing in expected-case $O(1)$ time. This approach requires $O(n^2)$ time and $O(n^2)$ space per edge in the worst case to generate $O(n)$ adjacent empty triangles. Thus the overall running time for both of the inner **FOR** loops in Step 4 is $O(n^4)$.

Unfortunately, if only a constant number of edges change status for every completion of the inner **FOR** loop, then the outer **REPEAT-UNTIL** loop may iterate $O(n^2)$ times and it appears we have a total running time of $O(n^6)$. However, a careful implementation avoids this problem.[5] Intuitively, we save time because the tests in both Steps 4a and 4b terminate when a single instance quadrilateral or edge contradicts the condition. When this happens, we need only keep track of where we were in the search, so that in subsequent tests we continue where we left off. In other words, for each edge we maintain a pointer to the last triangle tested above and below the edge: that is, the pair of triangles for which e was first found to be locally minimal, also called the *certificate* for e. In a later test of edge e, we resume the search with the same pair of triangles. Thus we never test more than n^2 quadrilaterals (pairs of triangles) per edges. So the total number of tests for all iterations of all edges is $O(n^4)$. We thus have an algorithm requiring $O(n^4)$ time and $O(n^2)$ space in the worst case.

[5] The idea for this came from Jack Snoeyink and the *LMT*-skeleton algorithm of [2]. Snoeyink's implementation accomplishes the same result in the same asymptotic running time without using multiple passes.

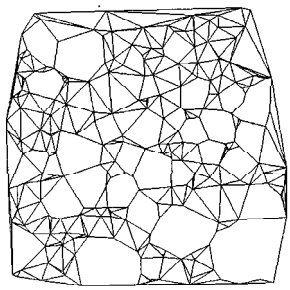

Fig. 3. *LMT*-skeleton of 250 uniform random points.

2.6. *Sample Skeletons*

We implemented our algorithms in Smalltalk/V on top of the existing objected-oriented workbench of [10]. We ran both the *LMT*-skeleton algorithm (Algorithm 1) and the *modified LMT*-skeleton algorithm on a number of different point sets. Figure 3 shows an example *LMT*-skeleton of 250 points taken from a uniform random distribution over a square region. In this example, all edges were found in three "rounds." (Step 4 was iterated three times to produce this graph. Nearly all edges were found in the first round.) Note that the subgraph is not only connected, it is close to being a complete triangulation, containing a very large percentage of the edges in *MWT*(*S*). The subgraph contains 607 edges, with only 312 edges remaining in *candEdges* as edges of unknown status. Of a total of 31,125 possible edges in $E(S)$, 30,206 edges were elminated. In a later section we present some more detailed experimental results. Figure 9 shows another *LMT*-skeleton of points in a "clustered" distribution.

Note finally that the remaining untriangulated regions in Fig. 3 are simple polygons with a small number of vertices. Thus the exact *MWT*(*S*) can be completed from this subgraph in $O(n)$ time. Specifically, we can determine in $O(n)$ time by depth-first search if the *LMT*-skeleton is connected. If it is, we search around each point to find untriangulated simple polygons, and triangulate each one in time proportional to the cube of the number of vertices. This step can be improved even further using a result of Bern and Eppstein [3] which shows that a simple polygon can be minimally triangulated in time $O(E_i^{3/2})$, where E_i is the number of *candEdges* in the untriangulated simple polygon i. (This follows from the fact that the number of empty triangles that can be formed with E edges is also $O(E_i^{3/2})$.) In the example of Fig. 3, the number of remaining *candEdges* in an untriangulated region is $O(n)$.

3. A Faster, More Extensive Modified *LMT*-Skeleton

With the use of Algorithm 1 and the *LMT*-skeleton, we are already producing the exact *MWT* of much larger point sets than have previously been computed for uniform random point sets. The next question is whether or not we can improve upon the *LMT*-skeleton, either in running time or in the number of known *MWT* edges produced. The answer is yes to both questions, at least in the expected case.

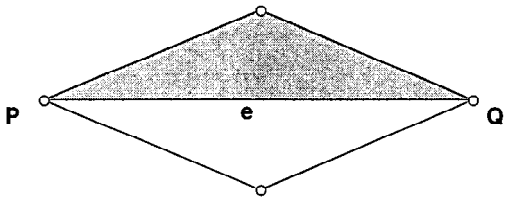

Fig. 4. Diamond property empty region.

Das and Joseph [8] have shown that all edges in the *MWT*(*S*) have what they call the *diamond property*.[6] This diamond property is as follows: for every edge *e* in *MWT*(*S*), consider the two triangular regions defined on either side of *e* such that *e* is the base of each triangle and the base angles are $\pi/8$, then at least one of the two triangular regions contains no other site in *S*. (See Fig. 4.)

This diamond property is a necessary but not sufficient condition for an edge *e* to be in *MWT*(*S*). The property alone does not produce a very large subgraph of known *MWT* edges, however, it can be used as a *pretest* to eliminate a large number of edges in an algorithm such as Algorithm 1. In fact, for a uniform distribution the expected number of edges that satisfy the diamond property is $O(n)$ [9]. For our modified approach, we initiallly place only those edges which do satisfy the diamond property pretest into *candEdges* and then run Algorithm 1. We call the resulting subgraph the *modified LMT*-skeleton.

Lemma 4. *LMT-skeleton* $\subseteq$ modified *LMT-skeleton* $\subseteq$ *MWT*.

Proof. (i) That *LMT*-skeleton $\subseteq$ *modified LMT*-skeleton follows directly from the algorithm: any edge *e* added to the *LMT*-skeleton in Step 4b will also be added to the *modified LMT*-skeleton in Step 4b because no new edges are introduced to *candEdges* that might intersect *e*.

(ii) The proof that *modified LMT*-skeleton $\subseteq$ *MWT* follows in the same manner as the proof of Lemma 3. By [8], we know that any edge failing the *diamond test* cannot be in the *MWT* and so its removal from *candEdges* cannot result in any edges being added that are not in the *MWT*. □

Note that the pretest may remove edges which *are* in some *other LMT*-skeleton (though not in the *MWT*), and thus the *modified LMT*-skeleton may contain some edges which are not in *every LMT*-skeleton. In fact, there are examples of point sets *S* where strict inequality holds and the *LMT*-skeleton(*S*) is a proper subset of the *modified LMT*-skeleton(*S*). Remember, however, that the primary purpose of the *LMT*-skeleton is to

[6] During the summer and fall of 1995 the authors of this paper had several conversations with Scot Drysdale and Scott McElfresh at Darmouth College, trying to prove a lemma for empty half-disks similar to the *diamond property* of Das and Joseph, with the goal of using it as a pretest in an *MWT* algorithm. Thanks to Scot Drysdale who eventually discovered the paper [8] and brought it to the attention of the authors (thus invalidating the need for the eventually proved half-disk property).

get a well-connected subgraph of the *MWT*; this modified approach is not only faster but produces at least as many known *MWT* edges as the original *LMT*-skeleton. Any extra edges are actually advantageous.

3.1. *Complexity of Modified Skeleton Approach*

For a uniform distribution, the expected number of edges passing the pretest is $O(n)$ [9]. Using bucketing, the pretest can be efficiently implemented in constant expected time per edge. So in the expected case, the initial set of *candEdges* is of size $O(n)$ and is computed in $O(n)$ time. The number of empty triangles that exist in a set of $O(n)$ vertices and $O(n)$ edges is $O(n^{1.5})$ [7], [3]. More importantly in the analysis of Step 4 is the following:

Lemma 5. *In a graph with n vertices and $O(n)$ edges, there are at most $O(n^2)$ quadrilaterals.*

Proof. Each quadrilateral is uniquely determined by a pair of opposite edges. (Two opposite edges give all four vertices of the quadrilateral.) Since there are only $O(n)$ edges, there can be only $O(n^2)$ pairs of edges. □

By Lemma 5, since the expected number of edges passing the diamond test is $O(n)$, the total number of empty quadrilaterals to be tested in Step 4 is $O(n^2)$. We generate the empty quadrilaterals by generating the empty triangles as in Algorithm 1, except we use a bucketing approach to test for emptiness in $O(1)$ time per triangle. As we saw earlier, the number of empty quadrilaterals is the determining factor in the number of iterations of Step 4. This suffices to prove the following:

Lemma 6. *The* modified *LMT-skeleton version of Algorithm* 1 *requires $O(n^2)$ time and $O(n)$ space in the expected case for a uniform random point set.*

4. Experimental Data

We implemented both the *LMT*-skeleton version of Algorithm 1 and the *modified LMT*-skeleton version using the *diamond pretest* described in Section 3. The implementation was done in Smalltalk/V for the Macintosh. We ran several thousand trials on various points sets up to 350 points using a collection of Power Macintosh machines running at 120 MHz with 32 MB RAM. For comparision, we also implemented algorithms for the 1.17682-skeleton and $\sqrt{2}$-skeleton (which are substantial faster to compute). Before presenting any numerical results, we first show some pictures of resulting subgraphs.

Figures 5 and 6 show the $\sqrt{2}$-skeleton and 1.17682-skeleton for the same random uniformly distributed point set as the *LMT*-skeleton of Fig. 3. Note that the 1.17682-skeleton is always a supergraph of the $\sqrt{2}$-skeleton and in general will contain more edges. However, the *LMT*-skeleton contains significantly more edges even than the 1.17682-skeleton.

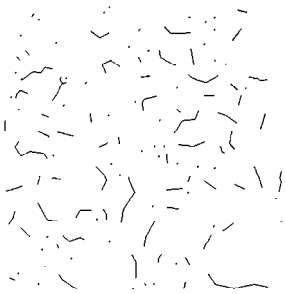

Fig. 5. sqrt(2)-Skeleton of the 250 points.

Figures 7 and 8 show the *LMT*-skeleton, and 1.17682-skeleton of another point set: 225 points taken in a uniform random distribution over a disk. Finally, Fig. 9 shows the *LMT*-skeleton of a different distribution of 250 points clustered in five groups of 50 points each at the corners and near the middle.

We now present some representative results in tabular form. Table 1 contains results from one set of 343 trials in which we computed the *LMT*-skeleton and the 1.17682-skeleton on the same sets of uniform random points ranging from 100 to 350 points. For the 50-point intervals, each line of data (each value of n) represents the averages of approximately 50 trials.[7] For the 1.17682-skeleton we report the average running time in seconds as well as the average number of edges. For the *LMT*-skeleton we report the running time, the number of edges in the skeleton, the number of remaining candidate edges (edges remaining in *candEdges* which may be in some *LMT* but are not necessarily in the *MWT*), and the number of edges eliminated. Since in every trial the *LMT*-skeleton was connected, we also completed the exact *MWT* and report the total number of edges in the triangulation as well as the total time to compute the exact *MWT*.

Note that in each of these 343 trials the *LMT*-skeleton was a connected graph, while in none of the trials was the 1.17682-skeleton connected. Since the 1.17682-skeleton is a

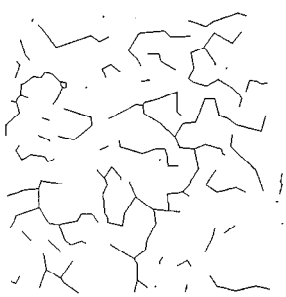

Fig. 6. 1.17682-Skeleton of the 250 points.

[7] For sizes of 125, 175, 225, and 275 the averages are for approximately 25 trials instead of 50, and because of the slower running times at 350 points, this average is taken over only seven trials.

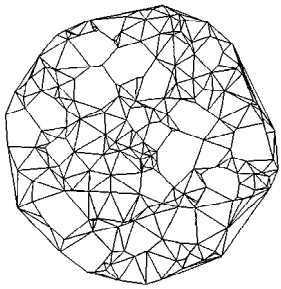

Fig. 7. *LMT*-skeleton of 225 points.

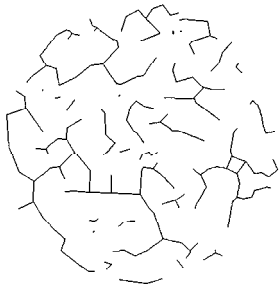

Fig. 8. 1.17682-Skeleton of 225 points.

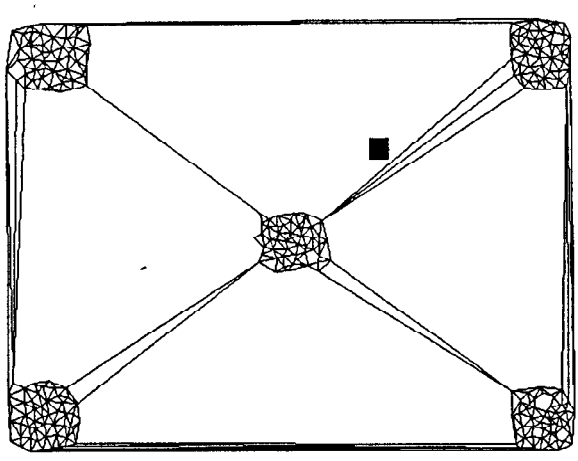

Fig. 9. *LMT*-skeleton of 250 “clustered” points.

Table 1. Subgraphs of the MWT.

	1.17682-Skeleton		*LMT*-skeleton				*MWT*	
n	Time (seconds)	Edges in	Time (seconds)	Edges in	Candidates	Edges out	Total edges	Time (seconds)
100	71	83	1,050	251	86	4,613	285	1,058
125	100	104	1,559	316	111	7,323	360	1,568
150	143	127	2,280	377	144	10,654	434	2,293
175	186	148	2,964	440	175	14,609	508	2,978
200	246	170	3,961	509	192	19,200	583	3,978
225	292	191	4,795	567	239	24,394	658	4,813
250	368	213	6,308	634	253	30,238	733	6,330
275	417	235	7,363	701	282	36,692	807	7,386
300	524	255	9,422	761	320	43,769	882	9,449
350	716	302	13,704	888	377	59,809	1,032	13,740

subgraph of the Delaunay triangulation, it can be computed very quickly (in $O(n \log n)$ time). However, the time required to complete the exact *MWT* using the method of [6] is $O(n^{k+2})$ where k is the number of unconnected components (after the convex hull is added). Even with the convex hull added, the 1.17682-skeleton of the point set in Fig. 6, for example, has 27 unconnected regions. To complete the exact *MWT*(S) using the algorithm of Cheng *et al*. [6] would require $O(n^{29})$ time, which is infeasible. On the other hand, computing the exact *MWT* from the *LMT*-skeleton—since it was connected—required just a few extra seconds (an average of 36 seconds for 350 points).

Finally, we mention that the modified version of Algorithm 1 making use of the diamond pretest as well as bucketing for both the pretest and empty triangle test resulted in a dramatic improvement in running time. For 250 points, our average run time dropped from 43,295 to 6308 seconds. The improvement in space complexity from $O(n^2)$ to $O(n)$ also allowed us to increase n significantly. This modified approach has also been implemented in *C* by McElfresh [17]. His results show a similar improvement over the *LMT*-skeleton approach. In one characteristic trial of 250 uniformly distributed points, the *modified LMT*-skeleton required approximately one-tenth the time of the *LMT*-skeleton. Belleville *et al*. have obtained even more dramatic results in their implementation described in [2] and subsequent improvements. Jack Snoeyink has communicated that with the pretest they are now running the algorithm on sets of size up to $n = 40{,}000$. It is interesting to note that even for these much larger sets, the resulting subgraph has been connected in every trial.

5. Conclusions and Discussion

The *LMT*-skeleton algorithm performs well. It requires $O(n^4)$ time and $O(n^2)$ space in the worst case. For uniform distributions of point sets ranging up to 350 points, it produced a connected graph in every trial, leading to a complete *MWT*. The result is a feasible algorithm to produce the exact *MWT* of uniform point sets in polynomial time. We also gave a modified version that runs faster, requiring only $O(n^2)$ time and

$O(n)$ space in the expected case, and producing more edges known to be in the *MWT*. However, this *modified LMT*-skeleton may contain edges that are not in all locally minimal triangulations.

5.1. *Unconnected Regions*

Although in all of our tests with *uniform* distributions the *LMT*-skeleton subgraph was connected, we have found distributions for which it is unconnected: a single point surrounded by a sufficient number of points on a circle, for instance, produces an *LMT*-skeleton that has one unconnected point. Bose *et al*. [4] have recently studied a similar case more closely, and given details of a construction of an 18-gon with a single point in the center, such that the *LMT*-skeleton of this configuration leaves the center point unconnected. They have also shown that such a configuration occurs with frequency linear in n for points in a uniform random distribution. However, the news is not as bad as it seems for the *LMT*-skeleton algorithm. First, even when this configuration of points occurs and the subgraph is not connected, it is still possible to complete the *MWT* efficiently. The dynamic programming approach suggested by Cheng *et al*. [6] requires $O(n^{k+2})$ time where k is the number of disjoint connected components. (When $k = 1$, we have the standard approach of [16] for the *MWT* of a simple polygon.) In the case of the configuration of [4], we have two connected components and so the *MWT* requires $O(n^4)$ with $n = 18$. Furthermore, though Bose *et al*. have given an elegant proof that this configuration actually appears with linear frequency, the constant of proportionality is very small. In fact, their result only shows that the expected number of disconnected components is larger than one when n is larger than 10^{51}. In other words, though appearing with "linear" frequency, it is unlikely to occur even once in any random point set that is ever likely to be tested.

Unfortunately, Belleville *et al*. [2] have extended the idea of [4] to show that the plane can be tiled with a configuration that does not lead merely to single disconnected points inside of 18-gons, but to a polygonal region with $2n/19 - o(n)$ holes and isolated points. Thus in this more troublesome example, the *MWT* cannot be directly completed from the *LMT*-skeleton in polynomial time using the method of [6] on the *LMT*-skeleton. This provides more persuasive evidence that the algorithm of this paper does not provide a worst-case polynomial-time solution to the *MWT* problem. Nevertheless, our algorithm performs very well on random point sets, and we have produced exact *MWT*s of much larger sets than have previously been produced. Also, there is still promise that the *MWT* of even this configuration can be computed efficiently because the number of remaining edges in *candEdges* is small.

5.2. *Extending Local Minimality*

Another possibility for increasing connectivity is to check larger areas for local minimality. The *LMT*-skeleton subgraph algorithm partitions $E(S)$ into sets of edges in all, no, or some locally minimal triangulation. We could create the same partition of the set of all possible triangles in S, or all possible quadrilaterals in S, etc., and similiar lemmas will

be true as long as we properly define "locally minimal." For instance, an empty triangle is locally minimal if when simultaneously flipping no two or three of its edges decreased its weight. Similarly, an empty quadrilateral is locally minimal if when simultaneously flipping no two, three, or four of its edges decreased its weight. Now, an algorithm similiar to the one we presented in this paper will work. Such an algorithm would still run in polynomial time, though much slower. Hence, it might be used as a second (and third, etc.) phase for any graph that is still disconnected after the *LMT*-skeleton subgraph algorithm is run. With each phase, the area of the regions being checked for minimality may increase, while the size of the regions that are still disconnected may decrease.

5.3. *Speeding Up the LMT-Skeleton Algorithm More*?

We also expect that the modified algorithm could be accelerated even further. Though the 1.17682-skeleton does not in general produce a connected graph, it is quickly computable ($O(n \log n)$ worst-case time) and produces a fair number of edges. As a preprocessing step, we might compute the 1.17682-skeleton, and add its edges to the initial *LMT*-skeleton, delete all edges in $E(S)$ intersecting those edges, and likewise delete triangles containing those deleted edges. We could then apply the diamond test to the remaining edges, and finally run Algorithm 1. This potentially gives a skeleton different from the *LMT*-skeleton, containing more potential edges, but one which is still provably a subgraph of the *MWT*.

We note that we could also explicitly generate and store the list *candTris* in $O(n^3)$ worst-case time and space using a method of Eppstein *et al*. for enumerating all empty triangles [12]. For worst-case space efficiency, we do not want to do this. We note, however, that for general point sets where all the points do not fall on the convex hull, the number of empty triangles will be much smaller than $\Theta(n^3)$ and this preprocessing may be useful. Rote [18] has implemented an output-sensitive variant of this algorithm using only $O(n^2 \log n + t)$ time and $(n + t)$ space, where t is the number of empty triangles produced. Also, the order in which the edges are tested in the **FOR** loops of Step 4 might have an effect as well.

5.4. *Other Dimensions*

One analog of the *MWT* problem in three dimensions is to find a tetrahedralization of a point set S that minimizes the sum of the weights of the triangles contained in it (where, again, the weight of a triangle might be its Euclidean surface area). An algorithm to find a subgraph of such a tetrahedralization could work similiarly to the *LMT*-skeleton subgraph algorithm. It would begin by enumerating all empty tetrahedra and triangles, and would then test each triangle for local minimality. We do not at present know how well such an algorithm would perform. One problem with this method, however, is that we do not even know how to find an *MWT* for a simple polyhedron in three dimensions. The naive dynamic programming algorithm does not work because adding one triangle does not partition the polyhedron as it does in two dimensions. So, even if we found a connected subgraph of an *MWT*(S), we have no way of patching the polyhedral holes in polynomial time. This is an interesting open problem.

5.5. *Open Problems*

Open problems stemming from the previous discussion include:

1. The complexity status of the *MWT* problem in the general case still remains open.
2. What is the worst-case disconnectivity of the *LMT*-skeleton?
3. How would the *LMT*-skeleton work in higher dimensions?

Acknowledgments

Thanks again to Scot Drysdale and Jack Snoeyink for helpful comments as mentioned earlier in this paper, and to Scott McElfresh for results from his implementation.

References

1. O. Aichholzer, F. Aurenhammer, M. Taschwer, and G. Rote, Triangulations intersect nicely. *Proc*. 11*th Annual Symposium on Computational Geometry*, 1995, pp. 238–247.
2. P. Belleville, M. Keil, M. McAllister, and J. Snoeyink, On computing edges that are in all minimum-weight triangulations. *Proc*. 12*th Annual Symposium on Computational Geometry*, 1996, pp. V7–V8; and *The* 5*th Annual Video Review of Computational Geometry*.
3. M. Bern and D. Eppstein, Mesh generation and optimal triangulations. In *Computing in Euclidean Geometry*, Ding-Zhu and F. Hwang, eds. World Scientific, Singapore, 1992, pp. 23–90.
4. P. Bose, L. Devroye, and W. Evans, Diamonds are not a minimum weight triangulation's best friend. TR 96-01, Dept. of Computer Science, University of British Columbia, January 1995.
5. S. Cheng and Y. Xu, Approaching the largest β-skeleton within the minimum weight triangulation. *Proc*. 12*th Annual Symposium on Computational Geometry*, 1996, pp. 196–203.
6. S. Cheng, M. Golin, and J. Tsang, Expected case analysis of β-skeletons with applications to the construction of minimum weight triangulations. *Proc*. 7*th Canadian Conference on Computational Geometry*, 1995, pp. 279–284.
7. N. Chiba and T. Nishizeki, Arboricity and subgraph listing algorithms. *SIAM J. Comput.* **14** (1985), 210–223.
8. G. Das and D. Joseph *Which Triangulations Approximate the Complete Graph*? Lecture Notes in Computer Science, Vol. 401, Springer-Verlag, Berlin, 1989, pp. 168–192.
9. M. Dickerson, S. Drysdale, S. McElfresh, and E. Welzl, Fast greedy triangulation algorithms. *Comput. Geom. Theory Appl*., to appear.
10. M. Dickerson, S. McElfresh, and M. Montague, New algorithms and empirical findings on minimum weight triangulations. *Proc*. 11*th Annual Symposium on Computational Geometry*, 1995, pp. 278–284.
11. M. Dickerson and M. Montague, A (usually?) connected subgraph of the minimum weight triangulation. *Proc*. 12*th Annual Symposium on Computational Geometry*, 1995, pp. 204–213.
12. D. Eppstein, M. Overmars, G. Rote, and G. Woeginger, Finding minimum area k-gons. *Discrete Comput. Geom*. **7** (1992), 45–58.
13. M. R. Garey and D. S. Johnson, *Computers and Intractability*; *A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.
14. P. D. Gilbert, New results in planar triangulations. Report R-850, University of Illinois Coordinated Science Laboratory, 1979.
15. J. M. Keil, Computing a subgraph of the minimum weight triangulation. *Comput. Geom. Theory Appl*. **4** (1994), 13–26.
16. G. Klincsek, Minimal triangulations of polygonal domains. *Ann. Discrete Math*. **9** (1980), 121–123.
17. S. McElfresh. Personal communication. St. Lawrence University, Work in progress.
18. G. Rote, Personal correspondence, 1996.
19. B.-T. Yang, Y.-F. Xu, and Z.-Y. You, A chain decomposition algorithm for the proof of a property on minimum weight triangulations. *Algorithms and Computation* (Conf. Proc., Beijing, 1994). Lecture Notes in Computer Science, Vol. 834. Springer-Verlag, Berlin, 1994, pp. 423–427.

Received May 29, 1996, *and in revised form January* 10, 1997.