Approximating the
Minimum Weight Triangulation

David Eppstein

Department of Information and Computer Science
University of California, Irvine, CA 92717

Tech. Report 91-55

July 2, 1991

Abstract

We show that the length of the minimum weight Steiner triangulation (MWST) of a point set can be approximated within a constant factor by a triangulation algorithm based on quadtrees. In $O(n \log n)$ time we can compute a triangulation with $O(n)$ new points, and no obtuse triangles, that approximates the MWST. We can also approximate the MWST with triangulations having no sharp angles. We generalize some of our results to higher dimensional triangulation problems. No previous polynomial time triangulation algorithm was known to approximate the MWST within a factor better than $O(\log n)$.

1 Introduction

Optimal triangulation has furnished a number of problems of longstanding interest in computational geometry. These problems have applications to cartography, spatial data analysis, and finite element methods. Optimization criteria for which efficient algorithms are known include maximizing the minimum angle [19, 23], minimizing the maximum angle [6], minimizing the minimum angle [7], minimizing the maximum aspect ratio [3], and minimizing the maximum edge length [5].

The most longstanding open problem in computational geometry is the complexity of another optimal triangulation problem, the *minimum weight triangulation* (MWT), in which the optimization criterion is the sum of the edge lengths. Indeed, this seems to have been known as the "optimal triangulation" for some time. The MWT problem is included in Garey and Johnson's famous list of problems neither known to be NP-complete, nor known to be solvable in polynomial time [8]. It is known that many triangulation algorithms will not correctly solve the MWT problem [16]. We do not resolve the status of this question.

The algorithms and problems cited above search for triangulations in which the vertex set of the triangulation is exactly the set of input points. Many of these problems can be extended to *Steiner* triangulation problems, in which the vertex set must be a superset of the input points. The additional vertices are known as Steiner points. For the application of triangulation problems to finite element mesh generation, Steiner triangulation is more natural than non-Steiner triangulation because the problem is simply to divide space up into a number of (triangular) cells, and extra cell boundary points are not a problem. However it is important that the number of additional Steiner points be relatively small, as this number directly affects the time to solve the resulting finite element system.

No Steiner triangulation problem is known to be exactly solvable in polynomial time, but a number of approximations have been published. In particular, one can find a Steiner triangulation in which all angles are bounded between 36° and 80°, using an algorithm based on *quadtrees* [1]. The number of Steiner points may vary with the geometry of the input, but the algorithm uses within a constant factor of the optimal number of points, and takes time polynomial in the total output size. Similar methods can be used to find a triangulation with $O(n)$ Steiner points in which no angle is obtuse [1]. We will revisit these two triangulations later.

For the minimum weight triangulation problem, it is not immediately clear that adding Steiner points can reduce the total edge length of the

MWT. But in fact we can show that the minimum weight Steiner triangulation (MWST) of n points can have a total weight $\Omega(n)$ times smaller than the MWT. Algorithms are known for approximating the MWT and the MWST [2, 15, 20, 24], but the best such algorithms have a total length that is $O(\log n)$ times the length of the MWT or MWST. In this paper we give the first constant-factor approximation algorithms for both the MWT and the MWST.

1.1   New Results

We describe algorithms for several triangulations, all based on the quadtree recursive space-partitioning data structure. Each of these triangulations has a total length that is $O(1)$ times the length of the MWST. Our algorithms can be implemented to take time $O(n \log n + k)$, where k is the number of Steiner points.

- We describe a simple Steiner triangulation based on quadtrees, in which each quadtree square is split until no square contains more than one input point, along with a balancing condition on the sizes of neighboring squares. This triangulation can use a nonpolynomial number of Steiner points, but is easy to implement and should perform well in practice. We prove that the total length of this triangulation is $O(1)$ times the MWST length.

- We modify the above algorithm to stop dividing squares below $O(\log n)$ levels in the quadtree. This reduces the number of Steiner points to $O(n \log n)$, while preserving the approximation to the MWST.

- We show that a variation of the algorithm from [1], in which some quadtree levels are "shortcut" to reduce the number of Steiner points to $O(n)$, and in which no triangle contains an obtuse angle, also approximates the MWST.

- We modify a triangulation from [1] which avoids sharp angles, and uses a number of Steiner points within a constant factor of the optimum needed to avoid sharp angles. The modified triangulation retains these properties and approximates the MWST. As in [1], a nonpolynomial number of Steiner points may be needed.

- We prove that the MWST can have a total weight that is $\Theta(\log n)$ times the weight of the minimum spanning tree (which we abbreviate MST; similarly the minimum weight Steiner tree would be the MSST).

2

The proofs of this and the following MWST properties are based on our result that quadtrees approximate the MWST.

- We prove that the MWT can have a total weight that is $\Theta(n)$ times the weight of the MWST.

Since minimum length triangulation has been motivated in part by finite element mesh generation, it is noteworthy that we can find triangulations which are good by other measures of their applicability to finite element analysis, and which also approximate the MWST.

The relation between MWT and MWST weight shows the importance of allowing Steiner points in a triangulation. The relation between MWST and MST weight is relevant because previous Steiner triangulation algorithms proved their performance by comparing the triangulation length to the MST; since any triangulation spans all vertices, and since the MST approximates the MSST [4], it follows that the MWST weight is at least a constant factor times that of the MST. However the result above shows that such a proof can never lead to an approximation factor better than $O(\log n)$. Our proof of is algorithmic: we find a point set for which our algorithms produce a triangulation with this length. The result then follows from the correctness of our approximation to the MWST.

1.2   Related Work

Quadtrees have been in use for well over a decade, with a number of applications [22] including finite element mesh generation [17, 21, 25]. The first theoretical analysis of quadtree triangulations appears in the recent work of Bern et al. [1] in which it is proved that such triangulations can be used to avoid sharp and obtuse angles, with a number of points within a constant factor of optimal. These restrictions on the angles in the triangulation are motivated by considerations from finite element analysis. We modify these algorithms slightly—the main difference is in a choice of several initial squares covering the points rather than a single large square—and show that the modified triangulations approximate the MWST. The unmodified algorithms turn out to be within an $O(\log n)$ factor of the MWST weight.

Previous attempts at approximating the MWT and MWST have followed two approaches. In the first approach, various workers have attempted to show that certain other standard triangulations approximate the MWT. For instance, it was at one point believed that the Delaunay triangulation actually achieved the minimum weight; however it has since been shown that it can be as far as $\Omega(n)$ from the optimum [10, 18]. This is pessimal since any

triangulation achieves $O(n)$ times the minimum weight [10]. Similarly, the greedy triangulation [9, 14] has been proposed as an approximation to the MWT; however the approximation factor can be as bad as $\Omega(\sqrt{n})$ [12, 18]. On the other hand, for convex polygons the greedy triangulation offers an easily computed approximation to the MWT [13, 14]; we use the greedy triangulation to relate the MWT and MWST for convex point sets.

The second approach to approximate MWT problems uses the insight that polygon minimum weight triangulation is significantly easier than the point-set MWT or MWST problems. The exact minimum weight triangulation of a simple polygon can be found by dynamic programming in time $O(n^3)$ [9, 11]. If the polygon is convex, a triangulation of weight $O(\log n)$ times the polygon's perimeter can be found by the *ring heuristic* of repeatedly connecting all pairs of adjacent even-numbered vertices [20], and as mentioned above an approximation to the MWT can be computed in linear time [13, 14]. By our new results, this is also an approximation to the MWST.

Lingas [15] suggested starting with polygonal regions formed by combining the convex hull with the MST of a point set, and then computing the optimal triangulations of these regions. He showed that this leads to a triangulation with a total length of $O(|MWT|\log n + nx\log n)$, where x is the length of the longest edge in the triangulation. However this formula does not show an approximation ratio better than $O(n)$.

Plaisted and Hong [20] used a more complicated method to partition the points into convex polygons. Then the ring heuristic can be used to triangulate the polygons, achieving a total triangulation length of $O(\log n)$ times the MWT length. The Plaisted-Hong algorithm has recently been implemented with a running time of $O(n^2 \log n)$ [24]. Plaisted and Hong conjecture that a version of their triangulation actually achieves a constant factor approximation to the MWT; this conjecture remains unproved. Since they do not use Steiner points, the Plaisted-Hong triangulation can be as far as $\Omega(n)$ from the MWST weight.

Clarkson [2] generalizes the ring heuristic to non-convex polygons, by allowing the addition of Steiner points. His method, together with Lingas' partition into polygons, achieves an $O(\log n)$ approximation to the MWST; until the present work this was the best such approximation known.

1.3   Organization of this Paper

In the next section we describe the simplest version of our quadtree algorithm, and prove that it approximates the MWST. The algorithm may have

nonpolynomial output size (number of Steiner points), but we show that the running time is polynomial in the input and output sizes. The proof of approximation first relates the triangulation to the MWT, and then shows that adding further Steiner points would only increase the length of the triangulation produced by our algorithm; therefore it also approximates the MWST.

In the third section we describe a number of modifications of our algorithm to produce triangles satisfying certain other properties. In particular we extend the algorithms of [1], in which all angles are within certain bounds, to new triangulations that both approximate the MWST and satisfy the angle bounds. We also show that $O(n)$ Steiner points suffice to approximate the MWST, therefore producing the first known polynomial time approximation to the MWST.

In the fourth section we prove a number of results about MWSTs, by analysing the behavior of the quadtree algorithm on various point sets. In the fifth section we introduce higher dimensional extensions of the MWT and MWST problems, discuss some difficulties in extending our results, and prove an $O(\log n)$-approximation to the minimum edge length Steiner triangulation in any dimension.

2 Basic Quadtree Triangulation

2.1 Triangulation Algorithm

A *quadtree* is a recursive partition of a region of the plane into axis-aligned squares. One square, the *root*, covers the region that is to be partitioned. Each square may be divided into four *child* squares, by splitting it with horizontal and vertical line segments through the center of the square. Each child has a size (length of the sides of the square) proportional to half the parent's size. Thus the collection of squares forms a tree, with smaller squares at lower levels of the tree. A *leaf square* is one that has not been further subdivided into children.

Each leaf square in the quadtree has a set of *neighbors*, those leaf squares sharing either corner vertices or portions of sides with the square. A neighbor is *orthogonally adjacent* if it shares a portion of the square's sides, and *diagonally adjacent* if it only shares a corner point.

As in [1], we maintain an additional *balance condition* in the quadtrees we construct: the orthogonally adjacent neighbors of any leaf square must be at most twice the size of the square, and at least half the size of the square. Equivalently, each line segment forming the boundary of a leaf square can
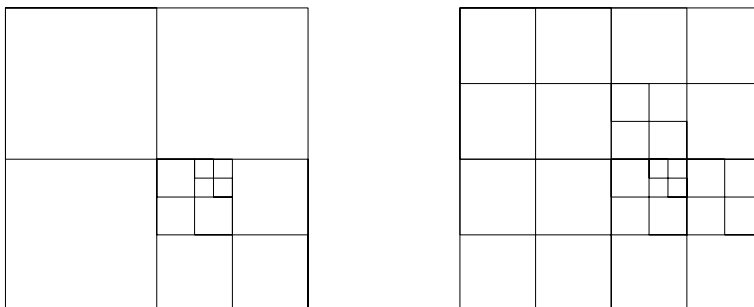
5

Figure 1. Balance condition for quadtrees: (a) unbalanced quadtree; (b) additional subdivisions to restore balance.

be divided into at most two parts by subdivision of neighboring squares. We say a quadtree is *balanced* if the balance condition is true. Unbalanced and balanced quadtrees are shown in figures 1(a) and 1(b).

Our triangulation algorithm first builds a quadtree covering the set of input points. Then it triangulates a rectangular region of the quadtree, including all line segments bounding leaf squares as edges in the triangulation. These edges are augmented by diagonal edges in squares not containing input points, and edges from the input points to the quadtree vertices surrounding them in the squares containing them.

The first phase of our triangulation is the placement of the root square. The main requirements are that the square cover all the input points, and that it not be too large. However to achieve a constant factor approximation we must be more careful than that. We first rotate the coordinate system so that the horizontal axis is aligned with the longest segment connecting any two input points (the long diagonal of their convex hull). Let x_1 and x_2 be the minimal and maximal extent of the points in the horizontal direction, and let y_1 and y_2 be the extent of the points in the vertical direction. We use a root square having a side length equal to the length of that long diagonal, and with a bottom side on the line $y = y_1$. The corners of the square are the four points (x_1, y_1), (x_2, y_1), $(x_1, (x_2 - x_1 + y_1))$, and $(x_2, (x_2 - x_1 + y_1))$. The placement of the root square is illustrated in figure 2(a); the long diagonal runs horizontally (because of the coordinate system rotation) and is indicated by a dashed line.

Next, we do some initial subdivision to reduce the area of the quadtree that we must actually triangulate. This step is essential in reducing the approximation ratio from $O(\log n)$ to $O(1)$. Let k be the largest integer for which $(x_2 - x_1)/2^k > (y_2 - y_1)$. Then we subdivide the quadtree k levels
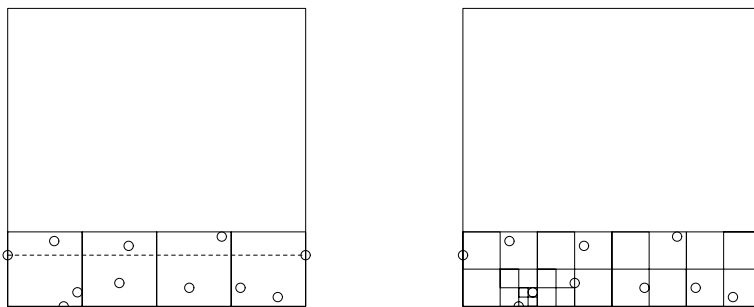
Figure 2. Quadtree triangulation: (a) placement and initial subdivision of root square, showing long diagonal; (b) subdivision until points are separated and squares are balanced.

deep, so that the leaf squares have size $(x_2 - x_1)/2^k$. By the choice of k, all input points will be contained in squares in the bottom row of the quadtree, so we only need triangulate this bottom row. The rest of the quadtree can be ignored for the remainder of the algorithm. The initial subdivision is also illustrated in figure 2(a).

We now further subdivide squares of the quadtree, maintaining the balance condition. We subdivide any square containing two input points in its interior or on its boundary. Whenever two neighboring squares violate the balance condition, we must also split the larger of the two squares, regardless of whether it contains an input point. At the end of this process, each input point is alone in its square. However this phase of the algorithm is inherently nonpolynomial, because two close together points may require a number of subdivisions unrelated to the total number of points. The result of this subdivision process is depicted in figure 2(b); in the figure only one square has been split because of the balance condition, but in practice this situation may arise more frequently.

Finally, we triangulate the resulting quadtree (i.e. the point set formed by the corners of the quadtree squares, together with the input points; our triangulation will include as edges the sides of each square). Each leaf square has between 4 and 8 quadtree vertices on its boundary, because of the balance condition. Each square may also contain an input point in its interior or on its boundary. Therefore there are at most 9 points to be triangulated, and no matter how we choose to perform this triangulation the resulting weight will be proportional to the size of the square.

Note that the order in which subdivisions are performed does not change the final quadtree. Therefore adding additional points to the input set

7

(within the original convex hull) can only increase the total amount of sub-division and therefore the total length of the triangulation.

2.2 Implementation Details

A number of details are required to perform the above triangulation algorithm efficiently; these are essentially the same as those required in [1].

In the course of the algorithm we maintain a partially split quadtree. For each square we maintain a data structure with pointers to its parent, its children, and its same-size neighbors. Whenever we split a square, we update the parent and child information. If the new children have neighbors of the same size, the neighbors are either siblings, or they are children of the neighbors of the split square. Therefore we can also maintain the neighbor pointers in constant time per subdivision.

To maintain the balance condition, we keep a queue of unbalanced neighbor pairs. Whenever we split a square, we determine if any of its parent's unsubdivided neighbors is a neighbor of the split square; if so the neighbor and some child of the split square form an unbalanced pair. This check can be performed in constant time, and it is not hard to see that all unbalanced pairs will be detected in this way. Whenever this queue is non-empty, we pull the top pair off the queue and check if the two squares still violate the balance condition; if so we subdivide the larger square and update the queue as necessary. By emptying the queue before continuing with the algorithm, we perform subdivisions to restore the balance condition before we allow any subdivisions for other reasons.

There are two ways to perform the checks that each input point is alone in its square. The simpler of the two is to add the points one at a time to the quadtree. Each leaf square contains a pointer to the point it contains, or is noted as containing no point. When we add a new point, we trace through the quadtree to find the appropriate leaf, and test if the leaf is empty. If not, we have a pair of points in the same square, and we subdivide further until the points are separated. This method will work for the truncated quadtree triangulation described later, but it can be inefficient for other quadtree algorithms (including the basic quadtree triangulation described above) because we may have many points at a low level in the tree, and each point takes time proportional to its level.

The second method for managing the input points is to deal with them all at once. Initially, the root square contains a list of all the points. When a box is split, we must divide that list into four parts, one for each of the child boxes. The difficulty here is performing that division efficiently when

the points are unevenly divided among the children. We solve this difficulty by maintaining two doubly linked lists, of the points sorted by horizontal and vertical coordinates. We split the square horizontally by moving in simultaneously from both ends of the horizontally sorted list, until we find the division corresponding to the center line of the square. We determine which of the two sides contains the smaller set of points, and we extract the points in that set one at a time from the vertically sorted list.

This takes time proportional to the size of the smaller set. However it leaves the vertical list of the smaller set unsorted. Along with the two lists, we maintain for each point two integers representing its order in the two lists. Whenever we extract a smaller subset of points from a larger set, we sort the smaller subset using the previously computed integers, then we recompute the integers in the smaller subset to be exactly the ranks in the sorted lists. However we do not recompute the integers for the points remaining in the larger subset until the size of the subset drops below half of what its size was the last time we performed such a recomputation. In this way the ranking integers can be maintained at a total cost of $O(n \log n)$ over the span of the algorithm. We use these ranking integers to sort the vertical set. If there are x points in the large set, the points will have ranks between 1 and $2x$. If we wish to extract k points, then the sorting can be performed in time $O(k \log(x/k))$ using a combination of bucket sorting (with k buckets) and comparison sorting. The total cost of splitting squares horizontally then becomes also $O(n \log n)$. Vertical splitting is performed in a similar way.

The initial choice of root square, and splitting into a bottom row of squares fitted to the vertical extent of the points, can be performed in time proportional to the number of squares on that bottom row.

Theorem 1. *A quadtree triangulation on n input points, with k Steiner points, can be constructed in time $O(n \log n + k)$*

Proof: The number of Steiner points is proportional to the number of times quadtree squares are subdivided. As described above, the total time to split a square is $O(1)$, together with the cost of splitting the sets of points in the square which can be amortized to a total of $O(n \log n)$. So if the total number of squares produced is k, the total time will be $O(n \log n + k)$. □

2.3   Approximation to MWT

We first show that the quadtree triangulation approximates the minimum weight (non-Steiner) triangulation of the input points. This is strange, be-

cause we are approximating a non-Steiner triangulation by a Steiner triangulation. However as we shall see this step is needed in our proof that the quadtree triangulation approximates the MWST.

Lemma 1. *The total edge length of the quadtree triangulation is $O(m)$, where m is the total edge length of the MWT.*

Proof: As we saw, the diagonals added to fill the quadtree out to a triangulation add a total length proportional to the lengths of the squares containing them. Therefore we need only add the perimeters of all leaf squares and show that this sum is proportional to the MWT length. We allocate each such square a charge proportional to its perimeter; we then reallocate this charge in a sequence of phases until all charge has been assigned to MWT edges. If the resulting charge on each MWT edge is proportional to the length of each edge, the quadtree triangulation must approximate the MWT.

It is not hard to show that the leaf squares containing points add to a total weight proportional to that of the minimum spanning tree, which *a fortiori* approximates the MWT. However we must also deal with leaf squares formed from maintaining the balance condition, and empty squares formed when the parent square contains multiple points, none of which happen to fall in one of the child squares.

Instead of summing the perimeter of leaf squares, we sum the perimeter of all quadtree squares containing at least one point. We can charge the perimeter of the leaf squares to this new sum as follows. Leaf squares containing points are charged to themselves. Squares for which a parent contains a point are charged to that parent. Any remaining squares must have been formed as a result of the balance condition, and for each such square one of the neighbors of the parent must contain an input point. We charge that neighbor. In this way each square is charged for the weights of at most 19 smaller squares (namely, three of its children, and four children each for its four same-size orthogonal neighbors). Therefore the total charge is proportional to the sum of the charged squares' perimeters.

We next find those squares in which all four child squares contain points, and divide the parent's charge evenly among the child squares. Each child will be charged at most half its perimeter for its parent, a quarter of its perimeter for its grandparent, and so on, so its charge will at most double. At this point we have reduced the problem to one of summing the squares which contain points and which have an empty child. The leaf squares cannot have their sole input point in the interiors of more than one quarter-
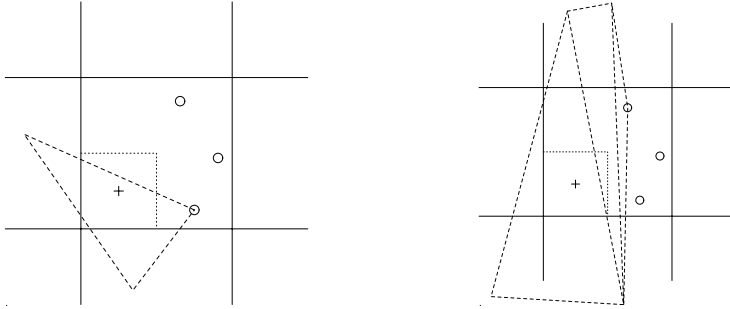
Figure 3. Triangle containing empty child center point: (a) one vertex is nearby; (b) moving across diagonals leads to a nearby vertex.

square, and so can be taken as having three empty children; alternately, they can be charged to their parents.

We now divide the problem into cases. The first case consists of squares entirely contained within the convex hull of the input points. Consider one such square s, with side length ℓ. Then the MWT partitions s into regions corresponding to the triangles in the MWT. By assumption s contains an empty child. Let (p_i, p_j, p_k) be the MWT triangle containing the centerpoint of this child. Since the child is empty, at least two of the sides of the triangle must have length at least $\ell/\sqrt{8}$. There are two subcases. If one of the triangle corners is within distance $\ell/2$ of s, charge the perimeter of s to the longest adjacent triangle edge (figure 3(a)). Otherwise, one edge of the triangle partitions s into two regions, one containing the centerpoint of the empty child and the other containing the input points in s. Consider the MWT triangle on the other side of this edge; since the triangle must be empty it again either divides the child's centerpoint from the input points in s, or it has one of the input points as its third vertex. In the latter situation, we charge either MWT edge touching that third vertex. In the former situation, we again test if the third vertex is within distance $\ell/2$ of s, and if not repeat the process; when we finally reach a nearby vertex we can charge a long adjacent edge (figure 3(b)). Each MWT edge is charged by at most 4 nearby squares of each possible size. The perimeter of a square charging an edge is at most $8\sqrt{2}$ times the edge length, and the possible sizes decrease in geometric series. Therefore each MWT edge is charged at most $64\sqrt{2}$ times its length for the squares in this case.

The second case consists of squares crossed by the convex hull boundary, but for which a same-size neighboring square is entirely contained in the convex hull. If the neighboring square contains a point, we charge the boundary
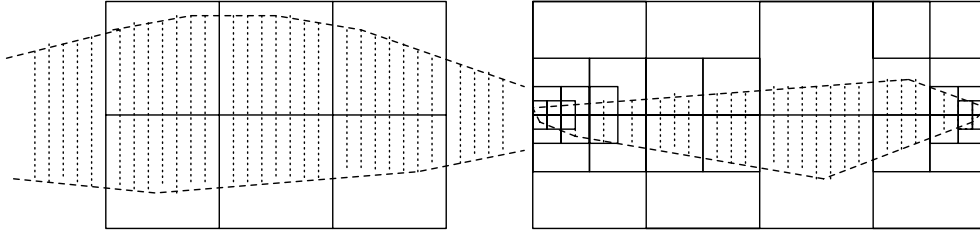
Figure 4. Squares with no neighbor interior to CH: (a) five neighbors, with CH boundary passing horizontally through each; (b) squares of each size form rows near the left and right sides of quadtree.

square to the neighboring square; each interior square is so charged at most 8 times. Otherwise, the neighboring square is empty. Again, we can move from the triangle containing its centerpoint to a long MWT edge with an endpoint near the original square. As in the first case, each MWT edge is charged proportionally to its length.

In the final case, we must account for squares crossing the convex hull boundary, for which all neighbors also cross the convex hull boundary. Because of the initial choice of quadtree position, there $O(1)$ such squares of each possible size in which the convex hull boundary passes through the top or bottom of the square. For the remaining squares, the boundary in the square and all of its neighbors passes through the left and right sides; therefore the squares of a given size have five similar neighbors, either up or down and to both the left and right (figure 4(a)). The three remaining possible neighbors are entirely outside the convex hull. Thus the squares of a given size in this case form double rows, above and below the long diagonal of the input points (figure 4(b)). If the initial squares have side length s, and the length of the long diagonal is r, then because s was chosen as small as possible, the squares of side length $s2^{-k}$ can only occur within distance $r2^{1-k}$ of the left and right sides of the quadtree. Therefore the total perimeters of all boxes in this final case is $O(r)$, which is proportional to the convex hull perimeter. Since each convex hull edge is in the MWT, this third case also can be charged to the length of the MWT. □

2.4   Approximation to MWST

We have shown that the quadtree triangulation approximates the MWT. But our desired result is that it approximates the MWST.
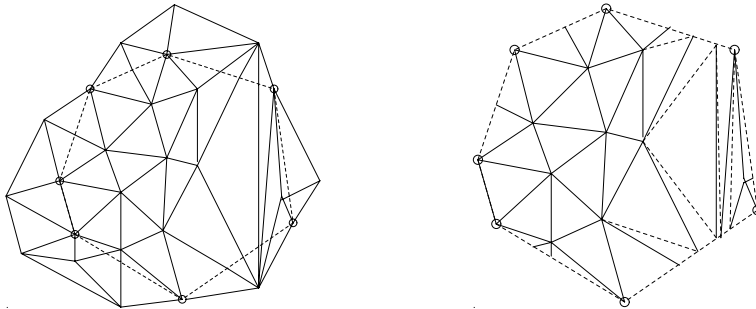
Figure 5. Triangulation within convex hull: (a) overlay hull boundary on MWST; (b) remove outside edges and retriangulate cut triangles.

Lemma 2. *Let T be a Steiner triangulation of a point set such that T is entirely contained within the convex hull of the input points, and having minimum weight over all triangulations satisfying that constraint. Then the weight of T is at most three times that of the MWST.*

Proof: Start with the $MWST$ of the input points, and overlay the convex polygon formed as the boundary of the input points' convex hull (figure 5(a) illustrates this process, however the triangulation depicted is not the MWST). Remove all edges outside the overlayed polygon. This will leave a partition of the interior of the convex hull into polygons. Each polygon is formed by intersecting a MWST triangle with the convex hull. Since the convex hull vertices are also vertices in the MWST, each polygon is simply an MWST triangle with up to three corners cut off. We then add $O(1)$ edges per cut triangle to form a new triangulation T' (figure 5(b)).

The edges of T' fall into three groups. The portions of MWST edges obviously have total length at most that of the original MWST. The edges formed by the overlayed convex hull boundary have length equal to the perimeter of the convex hull, which is less than or equal to the perimeter of the larger area covered by the MWST. The remaining group is the edges added to transform each cut MWST triangle into a collection of triangles in T'. We choose those edges in such a way that their length can be charged by the triangle inequality to disjoint portions of the boundary of the original triangle. The only way this could not be done would be if a triangle with three corners cut off (leaving a hexagon) were triangulated by three edges meeting at a single vertex (figure 6(a)). But such a hexagon can instead be triangulated by adding three edges forming a central triangle, in which case the edge length can be charged to the three sections of the original MWST
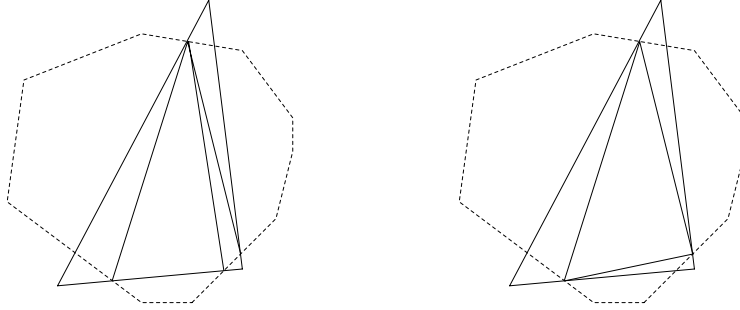
13

Figure 6. MWST triangle with three corners cut off by convex hull: (a) retriangu-
lation cannot be charged to triangle boundary; (b) improved retriangulation

triangle boundary outside the central triangle (figure 6(b)).

Therefore each of the three groups total at most the MWST weight, and so the whole triangulation T' totals at most three times that weight. The true minimum T satisfying the constraints has weight at most equal to T', and so is also within a factor of three of the MWST weight. □

Theorem 2. *The total edge length of the quadtree triangulation is $O(m')$, where m' is the total edge length of the MWST.*

Proof: Consider the MWST of an input point set. By lemma 2, we can assume without loss of generality that it is contained in the convex hull of the input points. Then it must be the MWT of its vertices, and the quadtree triangulation of the MWST vertices must approximate the MWST weight. But the root boxes of the MWST and of the original input points are identically placed, and adding Steiner points only increases the total leaf square perimeter of the quadtree triangulation. Therefore the quadtree triangulation of the original points has weight smaller than that of the MWST points, and so also approximates the MWST. □

3   Modified Quadtree Triangulations

3.1   Truncated Quadtree Algorithm

The quadtree triangulation above is unsatisfactory as an approximation to the MWST, because it does not run in polynomial time. This difficulty arises because input points may be very close to each other, and so many levels of the quadtree may be constructed before the points are separated.

14

Hence the running time may depend on the geometry of the input point set as well as on the number of points.

However, if two points are close together, the edges between them will contribute little to the total weight of the MWST, or to the weight of the quadtree triangulation. To make this precise we prove the following lemma.

Lemma 3. *If we consider a quadtree square of side length s, containing k input points, and replace its contents with any triangulation of the points together with the corners of the square and some of its side midpoints, the difference in weights between the portion of the original triangulation contained in the square, and the new triangulation in the square, is $O(ks)$.*

Proof: This follows from the fact that the total length of the edges in the replacement triangulation must be $O(ks)$. □

Therefore, from the lemma, if the root square length is r, if we modify the quadtree triangulation by not subdividing any square smaller than size r/n, we will achieve a total increase in length of $O(r)$. in the unsubdivided squares, together with some possible decrease due to fewer balance subdivisions elsewhere in the quadtree. But any triangulation must have total length $O(r)$, because that is proportional to the convex hull perimeter of the input points.

Theorem 3. *The truncated quadtree triangulation described above can be constructed in time $O(n \log n)$, has $O(n \log n)$ Steiner points, and approximates the MWST.*

Proof: If we stop subdividing when the squares reach size r/n, the quadtree will have at most $O(\log n)$ levels. Since the squares at each level contain points, are neighbors of squares containing points, or are children of such neighbors, the total number of squares at each level can be shown to be $16n$. Therefore the total size is $O(n \log n)$. Small squares containing several points can be triangulated using a plane sweep algorithm, in total time $O(n \log n)$. The same argument as for the original quadtree triangulation establishes the time bound for the rest of the construction. □

3.2 No Sharp Angles

We next examine the triangulation from [1], in which all angles are bounded between $36°$ and $80°$. This triangulation is based on quadtrees as before, with a few notable differences.
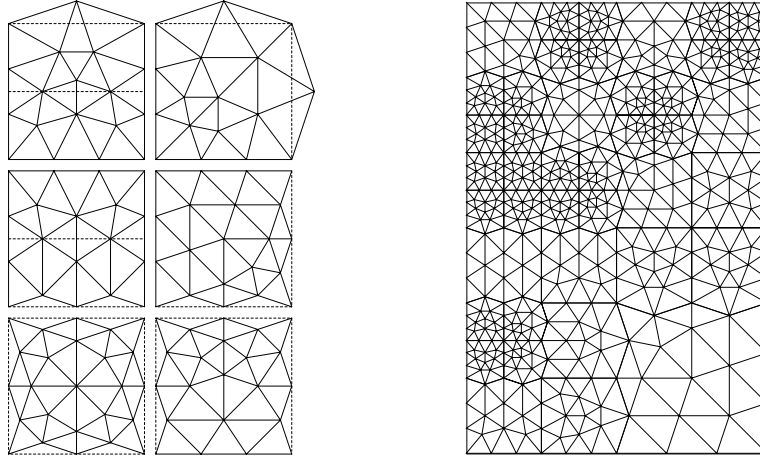
15

Figure 7. Angle-bounded triangulation: (a) tiles for triangulating empty squares; (b) example triangulation.

- The balance condition is stronger: diagonal neighbors as well as orthogonal neighbors are required to be within a factor of two in size from each other.

- Input points are *well separated* from each other: there must be a certain number of empty squares between them.

- Input points are well separated from the boundary of the quadtree; the quadtree root is chosen so that, if the root is divided into sixteen squares, all points are contained in the central four squares.

- Empty squares are triangulated by replacing them with one of a certain set of *tiles*; the choice of tile is made by examining the relative sizes of the four orthogonal neighbors. Tiles achieving the angle bounds above are shown in figure 7(a); an example triangulation is shown in figure 7(b).

- The squares of the quadtree are rearranged near input points so that each point is positioned close to the center of its square; then the square can be triangulated by an appropriate tile, using the input point as one of the vertices.

The strengthened balance condition can be handled by charging empty squares to farther away squares containing points; this changes our analysis

16

by at most a constant factor. Similarly the triangulation by tiles and the rearrangement near input points multiplies the length by at most a constant factor.

The only difference that causes us any difficulty is well separation of points from each other and from the boundaries. Well separation from each other again means we must charge empty squares to farther away squares containing points. However our reduction from MWT to MWST relied on the separation rule leading to the same result no matter what order it is applied. We can deal with this problem by performing all separations between input points before dealing with separations involving Steiner points; then as before the addition of Steiner points can only increase the total length. Since the quadtree triangulation of the MWST vertices is only a hypothetical device used in the proof of approximation, there is no problem with algorithmically distinguishing the two kinds of point.

Finally, we must choose a root square or collection of root squares, satisfying the separation of input points from root boundaries needed by the algorithm. In the quadtree triangulations above, the root squares contain points exactly on their boundaries. Also, the analysis in [1] of the number of Steiner points depends on the root square having few points, which would seem to conflict with our goal of fitting the root squares to the convex hull of the point set.

Recall that in our original algorithm we chose a sequence of squares, parallel to the long diagonal of the points, with size within a factor of two of the height of the points measured perpendicular to the long diagonal. This choice was used in our analysis, because it limits the total number of squares that touch the convex hull boundary and for which all same-size neighbors also touch the boundary.

If the square size is too small, there will be many squares, even when only a few Steiner points are needed to achieve small angles in the triangulation. We expand the square size, so that there are at most n initial squares; for the same reasons as in the truncated quadtree triangulation, this will not hurt our approximation to the MWST. We then expand the square size slightly, so that the convex hull of the input points is well separated from the outside of the initial squares. Again, this will not hurt our approximation.

The result of this is an initial set of $O(n)$ squares having the properties needed for the bounded angle triangulation. The analysis in [1] together with the appropriately modified proof of theorem 2 give us the following result.
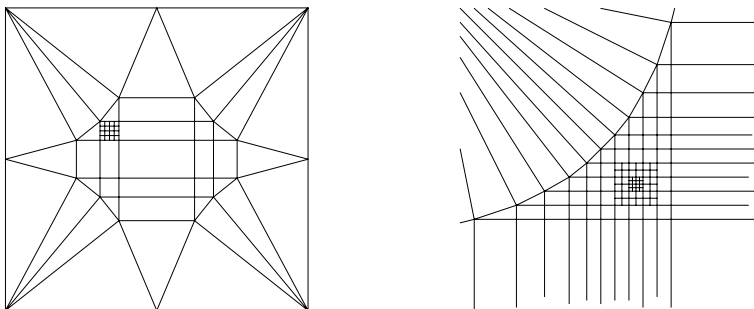
Figure 8. Attaching cluster to outer quadtree: (a) mirroring of cluster and attachment to outer square; (b) detail of attachment.

Theorem 4. *For any ϵ, let $k(\epsilon)$ be the minimum number of Steiner points required to triangulate a point set with no angle smaller than ϵ. Then we can find in time $O(n \log n + k)$ a Steiner triangulation with a total length $O(1)$ times the MWST length, in which all angles are bounded between $36°$ and $80°$, and for which the number of Steiner points is $O(k + n)$.* □

The area covered by the triangulation may be much larger than the convex hull of the input points; this is because of the expansion of the initial squares to reduce the number of initial squares to $O(n)$. We can forgo this expansion, at the expense of increasing the number of Steiner points. Without initial expansion, the number of Steiner points is $O(k' + n)$, where $k'(\epsilon)$ is the minimum number of Steiner points required to triangulate the points with no angle smaller than ϵ, and with total area proportional to the convex hull of the input set.

3.3 No Obtuse Angles

Bern et al. [1] describe another triangulation, in which $O(n)$ Steiner points are added and all angles are acute (strictly less than $90°$). Such a triangulation must be a Delaunay triangulation of its vertices. We now show how to modify this triangulation to approximate the MWST; again, the modification consists of judicious selection of the initial set of root squares.

The non-obtuse triangulation of [1] uses the insight that the only non-linear behavior of the quadtree triangulation occurs when a square containing a point set is repeatedly subdivided, without causing the point set to also be subdivided. This can happen because the points are all in the same child square, or because points are in different children but are not well sep-
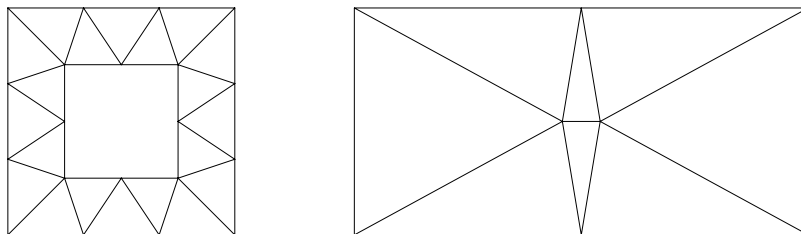
Figure 9. Extra tiles for non-obtuse triangulation: (a) connection to cluster; (b) acute triangulation of rectangle.

arated. If this happens for the same point set at more than some constant number of levels, we can identify a closely grouped *cluster* of points, which are causing the difficulty, and which are well separated from any other points in the input set. We then *shortcut* the non-linear behavior of the quadtree by recursively triangulating the cluster, and connecting this recursively constructed quadtree to the outer quadtree.

The connection is formed by imbedding the cluster in a small mesh of squares; this prevents subdivision caused by the balance condition from reaching outside the mesh. By some local rearrangements of the outer quadtree squares, we can cause the cluster to be placed near the center of the square containing it. We then *mirror* the cluster, placing copies of it in three locations forming a rectangle within the outer square (figure 8(a)). The mirror images contain copies of the mesh of squares but not of the actual input points. We can then connect the recursive quadtree and its mirror images to the outer squares, with a collection of acute and right triangles (figure 8(b)). By slightly warping the rectangle of mirror images, all right triangles can be made acute. Squares not containing input points or clusters are triangulated with tiles as in the angle-bounded triangulation. A new tile is used to connect the rest of the triangulation with the squares containing clusters (figure 9(a)). The connection between clusters and their outer squares uses a constant number of edges, which can therefore be charged to the length of the outer square.

As in the angle-bounded triangulation, the strengthened balance condition and the requirement of well-separated points can be dealt with at a constant factor cost in the approximation constant. We could choose initial squares as in that triangulation, but in fact we can do better in this problem. Recall that, in our basic quadtree triangulation algorithm, we choose a set of initial squares aligned with the long diagonal of the input points and with side length proportional to the maximum distance of any point

from that long diagonal. Denote the number of initial squares by m. If $m < n$, the number of Steiner points formed by the corners of the initial squares will not interfere with our total bound. Otherwise, at most n of the squares can actually contain a point. For each such square, we also keep two empty squares on either side; the outer empty square will remain undivided by balance condition requirements. This gives us a collection of $5n$ initial squares; the remaining $m - 5n$ squares are merged into a set of at most n rectangles. Each rectangle will touch on either side an unsubdivided square, which (because of the replacement by tiles) will have two equally spaced points on its boundary. Thus the rectangle can be split into three narrow rectangles, each of which can be triangulated with acute triangles by adding Steiner points at the midpoints of each long rectangle side and two Steiner points in the interior of each rectangle (figure 9(b)).

Theorem 5. *We can find in time $O(n \log n)$ a Steiner triangulation in which all angles are less than $90°$, with a total length $O(1)$ times the MWST length, covering an area $O(1)$ times the convex hull of the input points, and for which the number of Steiner points is $O(n)$.*

Proof: The proof that angles are acute, that the number of Steiner points is $O(n)$, and that the time is $O(n \log n)$, are all the same as in [1]. The remaining difficulty in proving approximation to the MWST is that, by recursively triangulating clusters, we lose the property that adding Steiner points can only further subdivide the quadtree squares. The problem is that the Steiner points can cause the cluster to not be separated from the main quadtree, or cause the cluster to have a larger convex hull, in either case leading to squares with a different alignment.

We consider a hypothetical modification to the above algorithm. When we recursively triangulate a cluster, instead of fitting the root square around the cluster, we choose a root square (or set of up to four such squares) that could also be formed by sufficient subdivision of the original quadtree. In this modified triangulation, additional points can now only cause additional subdivision. If additional points cause a cluster to not be found, the squares used in triangulating that cluster will instead be part of the main quadtree. This modified triangulation can then be shown to approximate the MWST, exactly as in the proofs of the other triangulations above. Indeed, any square formed in the modified triangulation must also be formed in the angle-bounded triangulation (because we use the same balance and separation properties).

We then note that for each square (containing an input point) in the true non-obtuse triangulation defined above, we can find a nearby square
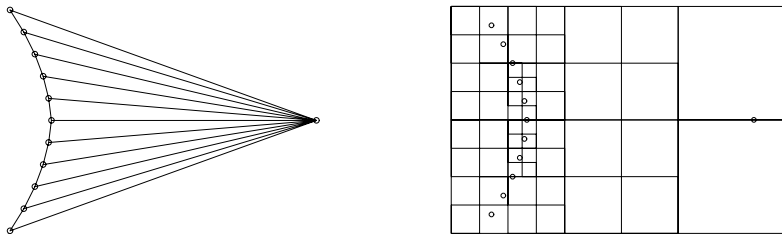
Figure 10. Point set illustrating ratio of MWT to MWST lengths: (a) MWT length $\Omega(yn)$; (b) quadtree length $O(y + x \log n)$.

(containing the same point) of similar size in the non-obtuse triangulation. Therefore the non-obtuse triangulation defined above also approximates the MWST. □

4 Properties of the MWST

4.1 MWST and MWT

Clearly the MWST must have total edge length at most that of the MWT. It is also not hard to show that the weight of the MWT (or any other non-Steiner) triangulation is $O(n)$ times the perimeter of the convex hull, and therefore $O(n)$ times the MWST weight. As we now show, this latter factor can be tight, even for the MWT.

Theorem 6. *For any n, there exist point sets for which the MWT has weight $\Omega(n)$ times the MWST weight.*

Proof: Start with an isosceles triangle, with two long side lengths equal to some value y and one short side length equal to x. Let the short side be replaced by a concave chain of $n - 1$ vertices, giving (with the remaining vertex at the apex of the triangle) a total of n. Then the only empty triangles touching the apex are those formed by pairs of adjacent vertices in the concave chain; therefore all such triangles must be present in the MWT, and the total MWT length is $\Omega(yn)$ (figure 10(a)).

However the quadtree triangulation will contain subdivided squares of size s only within within distance $O(s)$ of the concave chain. There can be at most $O(x/s)$ such squares, for a total length of $O(x)$ at each level of subdivision. As usual we need only count squares of side length $\Omega(y/n)$, for which there are $O(\log n)$ possible sizes. The total length of the quadtree triangulation, and hence the MWST, is $O(y + x \log n)$ (figure 10(b)).

21

If we let $x = y/\log n$ we achieve a ratio of $\Omega(n)$ between the MWT and MWST lengths as desired. $\square$

4.2 MWST and MST

The algorithm of Clarkson [2] shows that there exist Steiner triangulations with a total weight within an $O(\log n)$ factor of the MST weight. A similar result holds for our triangulations (see lemma 8 below). Therefore the MWST weight is within a logarithmic factor of the MST weight. We now show that this is tight.

Theorem 7. *For any n, there exist point sets for which the MWST has a weight $\Omega(\log n)$ times the MST weight.*

Proof: Let the points be equally spaced around the unit circle. Then the MST weight is approximately $2\pi(1 - 1/n) = O(1)$. The quadtree triangulation will produce squares of all sizes down to the level at which the points are separated from each other; this occurs when the side length nears $1/n$, at a level $\Omega(\log n)$. For each side length s, there will be approximately $1/s$ unsubdivided squares of that size, created because of the balance rule, in two chains running around the inside and outside of the unit circle. The total length of these squares is $\Omega(1)$. Therefore the total length of all unsubdivided squares, and the total length of the quadtree triangulation, is $\Omega(\log n)$. But since the quadtree triangulation approximates the MWST, the MWST length must also be $\Omega(\log n)$. $\square$

5 Extensions to Higher Dimensions

An obvious question arises from these results: do they carry over into higher dimensions? Can quadtree triangulations approximate the minimum weight triangulation of higher-dimensional point sets?

To answer this question, we first have to determine what we mean by the minimum weight triangulation. Given a triangulation, we can measure its 0-faces (points), 1-faces (edges), etc., up through d-faces. The number of 0-faces must be at least n, and any non-Steiner triangulation will have that many. The d-faces must have total measure at least that of the convex hull, and again any non-Steiner triangulation achieves this. That leaves $d-1$ possible functions to minimize. In two dimensions, the only interesting minimization problem is that of the total edge length. But even in three dimensions, one can examine both edge length and triangle area.

Note that in two dimensions, our triangulations not only optimize edge length, but at least one of them (the non-obtuse triangulation) also uses $O(n)$ Steiner points and fits within an area $O(1)$ times the area of the original point set. Thus it simultaneously optimizes all three possible criteria. However we saw that this simultaneous optimization must be modified if we desire further properties, such as no small angles. Then we must count the optimal number of Steiner points over triangulations also having these properties. Nevertheless it is reasonable to hope that similar simultaneous approximation results hold in higher dimensions.

Let us examine the three dimensional case in more detail. First we must determine what to use as the root box of our octtree (three-dimensional quadtree). If we wish to approximate the minimum triangle area, we must closely fit our initial set of cubes to the convex hull of the point set; otherwise the surface area of our initial set will be too large. On the other hand, if we start with an initial set of cubes that are too small, the total edge length would be too large. It seems that already in the initial placement we need some sort of hierarchical decomposition.

Next we examine our proof of optimality. The main step was amortizing cubes with an empty child against MWT edges with a nearby endpoint. We did this by starting at the triangle containing the child's centerpoint. If the endpoints were not nearby, one face of the triangle separates the centerpoint from the points in the square, and we can use that fact to move from there to a large triangle with a nearby vertex. In three dimensions, we can similarly reduce the problem to counting squares with an empty child. The tetrahedron containing the child's centerpoint will have sufficiently long edges, but it may be long and narrow, having a surface area that is arbitrarily small. Indeed, it may be that no triangle has area proportional to that of the cube. The triangle's edges will be long, but if the endpoints are not nearby it will not be clear in which direction to move in order to find a nearby triangle with long edges.

Given these difficulties, it is surprising that we can find any higher dimensional generalization of our results. But in fact we can prove the following, which shows that quadtree triangulations give a (non-constant-factor) approximation to the minimum edge length triangulation.

Theorem 8. *In any dimension d, the appropriate generalizations of the quadtree and truncated quadtree triangulations, starting from a single root box, use total edge length $\exp(cd) \log n$ times the length of the MST, for some constant c.*

Proof: As before, we can charge the length of boxes not containing points

23

to nearby boxes containing points. Each box is charged $exp(O(d))$ times its length. Then we need merely sum the edge lengths in boxes containing points. The boxes with edge length $1/n$ times the initial root box take a total length proportional to the root box length, so we need only look at higher levels in the quadtree. There are $O(\log n)$ such levels; we show that the length in each is proportional to the minimum spanning tree length.

Consider a box of side length s, containing a point x. There are $3^d - 1$ neighboring boxes. There are $O(3^d)$ boxes large enough that the entire point set is contained in these neighbors; the total length of these boxes is $exp(O(d))$ times the MST length. Otherwise, x is connected by the MST to a point outside the neighboring boxes; therefore there must be an MST path from x to that point, and the length of the intersection of that path with the neighboring boxes must be $O(s)$. We charge that portion of the path with the weight of the box; each portion of the MST is charged 2^d times its length for as many as 3^d boxes at each of the $O(\log n)$ levels. $\square$

As a corollary, d-dimensional quadtree triangulation approximates the minimum edge length Steiner triangulation within a factor of $O(\log n)$. This argument also applies in two dimensions, and proves that the triangulations of [1] (in which the root box was not fitted as closely to the input points as it is in our constructions) also achieve this $O(\log n)$ approximation.

We do not know any similar approximation results for the measures of k-faces in the triangulation, $k > 1$.

6   Conclusions

We have shown that quadtree triangulations achieve a small total edge length. This complements their previously known ability to restrict the angles used in the triangulation, and is further evidence of their utility in finite element mesh generation applications. We also used quadtree approximations to the MWST as a tool in proving other properties of the MWST.

Our analysis of the approximation ratio is quite sloppy and leads to a number in the hundreds. We believe the analysis can be tightened to lead to approximations on the order of ten times the MWT or MWST weight. Further improvements may come from tuning the algorithms in various ways, from heuristic procedures such as removing Steiner points that do not contribute to decreased length, and from performing various other local optimizations. It would be of interest to determine the exact constants for our algorithms, and for improvements to them.

Another important open question is whether it is possible to approximate

the MWT. At first glance, quadtrees seem unlikely to help, because they use Steiner points and can achieve much lower lengths than the MWT. Hence it would seem difficult to rearrange a quadtree triangulation into a non-Steiner triangulation that approximates the MWT. Plaisted and Hong [20] conjecture that their algorithm computes an approximation to the MWT, but it seems difficult to determine when their $O(\log n)$ factor lost due to the ring heuristic is a necessary part of the MWT, and when it is an artifact of their construction. Since in the MWST problem the same $O(\log n)$ factors sometimes appear, and are easily explained with our quadtree approximation, perhaps our quadtree based techniques can be used to solve this problem.

Acknowledgements

References

[1] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. 31st IEEE Symp. Found. Comp. Sci. (1990) 231–241. Some cited results appear only in the full version, submitted for journal publication.

[2] K. Clarkson. Approximation algorithms for planar traveling salesman tours and minimum-length triangulations. 2nd ACM-SIAM Symp. Discrete Algorithms (1991) 17–23.

[3] E.F. D'Azevedo and R.B. Simpson. On optimal interpolation triangle incidences. Report CS-88-17, Univ. Waterloo (1988).

[4] D.Z. Du and F.K. Hwang. An approach to proving lower bounds: solution of Gilbert-Pollack's conjecture on Steiner ratio. 31st IEEE Symp. Found. Comp. Sci. (1990) 76–85.

[5] H. Edelsbrunner and T.S. Tan. A quadratic time algorithm for the min-max length triangulation. Report UIUCDCS-R-91-1665, Univ. Illinois, Urbana-Champaign (1991).

[6] H. Edelsbrunner, T.S. Tan, and R. Waupotitsch. A polynomial time algorithm for the minmax angle triangulation. 6th ACM Symp. Comput. Geom. (1990) 44–52.

[7] D. Eppstein. The farthest point Delaunay triangulation minimizes angles. Report ICS-90-45, Univ. California, Irvine (1990).

[8] M.R. Garey and D.S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman (1979).

[9] P.D. Gilbert. New results in planar triangulations. Report R–850, Univ. Illinois Coordinated Science Lab (1979).

[10] D.G. Kirkpatrick. A note on Delaunay and optimal triangulations. Inf. Proc. Lett. 10 (1980) 127–128.

[11] G.T. Klincsek. Minimal triangulations of polygonal domains. Ann. Disc. Math. 9 (1980) 121–123.

[12] C. Levcopoulos. An $\Omega(\sqrt{n})$ lower bound for non-optimality of the greedy triangulation. Inf. Proc. Lett. 25 (1987) 247–251.

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

[14] C. Levcopoulos and A. Lingas. Fast algorithms for greedy triangulation. 2nd Scand. Worksh. Algorithm Theory, Springer-Verlag LNCS 447 (1990) 238–250.

[15] A. Lingas. Advances in minimum weight triangulation. Ph.D. thesis, Linköping Univ. (1983).

[16] E.L. Lloyd. On triangulations of a set of points in the plane. 18th IEEE Symp. Found. Comp. Sci. (1977) 228–240.

[17] R. Löhner. Some useful data structures for the generation of unstructured grids. Commun. Appl. Numerical Methods 4 (1988) 123–135.

[18] G.K. Manacher and A.L. Zobrist. Neither the greedy nor the Delaunay triangulation approximates the optimum. Inf. Proc. Lett. 9 (1979) 31–34.

[19] D.M. Mount and A. Saalfeld. Globally-equiangular triangulations of co-circular points in $O(n \log n)$ time. 4th ACM Symp. Comput. Geom. (1988) 143–152.

[20] D.A. Plaisted and J. Hong. A heuristic triangulation algorithm. J. Algorithms 8 (1987) 405–437.

[21] W.C. Rheinboldt and C.K. Mesztenyi, On a data structure for adaptive finite element mesh refinements. ACM Trans. Math. Softw. 6 (1980) 166–187.

[22] H. Samet. The quadtree and related hierarchical data structures. Computing Surveys 16 (1984) 188-260.

[23] R. Sibson. Locally equiangular triangulations. Computer J. 21 (1978) 243–245.

[24] W.D. Smith. Implementing the Plaisted-Hong min-length plane triangulation heuristic. Manuscript cited by [2] (1989).

[25] M.A. Yerry and M.S. Shephard. A modified quadtree approach to finite element mesh generation. IEEE Comp. Graphics and Appl. 3 (1983) 39–46.