Algorithmica (1987) 2: 137-151

Algorithmica
© 1987 Springer-Verlag New York Inc.

A Faster Divide-and-Conquer Algorithm for Constructing Delaunay Triangulations[1]

Rex A. Dwyer[2]

Abstract. An easily implemented modification to the divide-and-conquer algorithm for computing the Delaunay triangulation of n sites in the plane is presented. The change reduces its $\Theta(n \log n)$ expected running time to $O(n \log \log n)$ for a large class of distributions that includes the uniform distribution in the unit square. Experimental evidence presented demonstrates that the modified algorithm performs very well for $n \leq 2^{16}$, the range of the experiments. It is conjectured that the average number of edges it creates—a good measure of its efficiency—is no more than twice optimal for n less than seven trillion. The improvement is shown to extend to the computation of the Delaunay triangulation in the L_p metric for $1 < p \leq \infty$.

Key Words. Delaunay triangulation, Voronoi diagram, L_p metric, Computational geometry, Average-case complexity, Analysis of algorithms.

1. Introduction. The *Voronoi diagram* generated by a set $\mathcal{S} = \{P_1, P_2, \ldots, P_n\}$ of n points in the plane (called *sites*) partitions the plane into n convex regions $\mathcal{V}_i = \{P \mid \forall j\colon d(P, P_i) \leq d(P, P_j)\}$. Each region contains the points lying nearer to the site in its interior than to any other site.

The straight-line dual of the Voronoi diagram is called the *Delaunay triangulation*, denoted $\mathrm{DT}(\mathcal{S})$. P_i and P_j are adjacent in $\mathrm{DT}(\mathcal{S})$ if and only if $\mathcal{V}_i$ and $\mathcal{V}_j$ share a common edge (Figure 1.) If, as is assumed in the sequel, no four sites are cocircular, the Delaunay triangulation partitions the convex hull of $\mathcal{S}$ into triangles. The Voronoi diagram can be constructed from the Delaunay triangulation in $O(n)$ time and *vice versa*.

Several approaches have been taken to the design of algorithms for constructing the widely applied Voronoi diagram and Delaunay triangulation.

Shamos and Hoey [12] present the first divide-and-conquer algorithm for constructing the Voronoi diagram and show its $\Theta(n \log n)$ worst-case running time to be optimal under the real-RAM model of computation. Lee and Schachter [7] describe a dual algorithm for constructing the Delaunay triangulation. Guibas and Stolfi [3] advocate the Delaunay triangulation as an intermediate step in the construction of the Voronoi diagram, and present ideal data structures for the problem. Hwang [4], Lee and Wong [8], and Lee [6] present $O(n \log n)$ divide-and-conquer algorithms for the Voronoi diagram in the L_1, L_1 and L_∞, and general L_p metrics, respectively. Ohya *et al.* [11] show the average running time

[1] This research was supported by National Science Foundation Grants DCR-8352081 and DCR-8416190.

[2] Carnegie-Mellon University, Pittsburgh, Pennsylvania, USA.

Received May 14, 1986; revised October 22, 1986. Communicated by Bernard Chazelle.

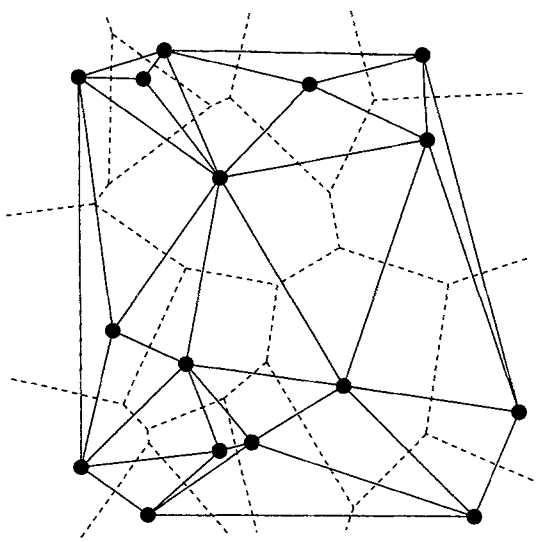
Fig. 1. A Voronoi diagram and its dual.

of these divide-and-conquer algorithms to be $\Omega(n \log n)$ when the sites are uniformly distributed in the unit square.

Fortune's [2] sweepline algorithm uses a clever geometric transformation to achieve $\Theta(n \log n)$ worst-case time. Only the trivial bounds on its average performance are known.

Various incremental algorithms, which construct the Voronoi diagram by adding new sites one by one, require $\Theta(n^2)$ time in the worst case, since a new site may be a neighbor to all previously inserted sites. Such methods differ principally in the order in which sites are added. Sibson and Green [15] were early advocates of such a method, giving an algorithm with $O(n^{3/2})$ average running time. The algorithm of Ohya *et al.* [11] attains optimal linear average time.

Algorithms of another variety construct the Delaunay triangulation triangle by triangle. McLain's algorithm [9] requires $\Theta(n^2)$ time in the worst case. An improvement described by Maus [10] requires only linear time on the average.

Finally, Bentley *et al.* [1] propose a complicated hybrid algorithm. It invokes a divide-and-conquer algorithm on the "outer" sites lying near the boundary of the unit square. The Voronoi polygon of each "inner" site is constructed in constant expected time by a "spiral search" among its neighbors. In the unlikely event that more than $O(\log n)$ time is expended on any inner site, spiral search is abandoned and the divide-and-conquer algorithm is applied to the entire set of sites. While achieving asymptotically optimal performance in both the worst and average cases, its implementation is complicated due to its use of two distinct methods.

We present an easily implemented improvement to the divide-and-conquer algorithms, including those for the L_p metric for $1 < p \leq \infty$. If the floor function can be computed in constant time, the change maintains its $O(n \log n)$ worst-case complexity, but lowers its average-case complexity to $O(n \log\log n)$ when the sites are drawn independently from any of a large class of distributions which includes the uniform distribution on the unit square. Our experimental evidence demonstrates that in the Euclidean metric the improved algorithm performs very well for $n \leq 2^{16}$, the range of the experiments. We conjecture that the average number of edges it creates—a good measure of its efficiency—is no more than twice optimal for n less than seven trillion.

2. Preliminaries.

We now restate some useful facts about Delaunay triangulations and Voronoi diagrams given by Lawson [5], Shamos and Preparata [13], and Guibas and Stolfi [3]. We take Guibas and Stolfi's very complete exposition as our point of departure and often speak only in terms of their algorithm. In fact, our results extend to the other divide-and-conquer algorithms without difficulty.

LEMMA 2.1. *Every triangulation of a set of n sites of which k lie on the convex hull (i.e., on the boundary of the convex hull) has* $3n-k-3$ *edges and* $2n-k-2$ *triangles.*

PROOF. Let t be the number of triangles and e the number of edges. All the edges belong to two triangles except for the k on the convex hull, thus $2e-k=3t$. The results are obtained by solving this equation for t or e and substituting into $n-e+(t+1)=2$ (Euler's formula). □

COROLLARY 2.2. *At most* $3n-6$ *nonintersecting edges can be constructed on a set of n sites.*

PROOF. Any set of nonintersecting edges can be extended to form a triangulation. □

LEMMA 2.3. *If sites* P_i, P_j, *and* P_k *are vertices of a triangle in the Delaunay triangulation, the circle passing through these three sites contains no other sites.*

PROOF. The center of the circle is equidistant from the three sites and is on the boundary of the Voronoi region of each of them. Thus it cannot lie nearer to a fourth site than to P_i, P_j, and P_k. □

COROLLARY 2.4. *If sites* P_i *and* P_j *are endpoints of a Delaunay edge, there is some circle containing no sites which passes through* P_i *and* P_j.

PROOF. Each edge is a side of some triangle. □

COROLLARY 2.5. *Every convex-hull edge is a Delaunay edge.*

PROOF. One of the half-planes defined by the line through the endpoints of the edge is a degenerate site-free circle. □

Lawson [5] and Sibson [14] have showed that, degenerate cases with four cocircular sites excepted, the Delaunay triangulation is the only triangulation in which every triangle satisfies Lemma 2.3.

Guibas and Stolfi's algorithm first constructs the Delaunay triangulation. The Voronoi diagram is then found in linear time. The Delaunay triangulation is constructed as follows:

1. The sites are sorted by increasing x-coordinate.
2. If there are three or fewer sites, the Delaunay triangulation is constructed directly. Otherwise the sites are divided into two approximately equal sets by a vertical line, step 2 is recursively applied to construct the Delaunay triangulations of these sets, and the results are merged.

The merge procedure forms the foundation of our own algorithm as well as that of Guibas and Stolfi. Let l be the dividing line of step 2, and let $\mathscr{L}$ and $\mathscr{R}$ be the sets lying to the left and the right of l. Clearly, two edges of the convex hull of $\mathscr{L} \cup \mathscr{R}$ cross l. Merging begins with a search for the endpoints of the lower of the two. The search begins with the sites of $\mathscr{L}$ and $\mathscr{R}$ lying nearest l and alternately advances clockwise around the convex hull of $\mathscr{L}$ and counterclockwise around the convex hull of $\mathscr{R}$. Once its endpoints are found and the lower hull edge is created, the other new edges are created in the order in which they cross l. Old edges are removed if intersected by a new edge. The essential features of the merge procedure are summarized in the following theorem:

THEOREM 2.6. *Let l, $\mathscr{L}$, and $\mathscr{R}$ be as above. When merging* $\mathrm{DT}(\mathscr{L})$ *and* $\mathrm{DT}(\mathscr{R})$ *to construct* $\mathrm{DT}(\mathscr{L} \cup \mathscr{R})$:

(a) *Only edges joining two sites in $\mathscr{L}$ and edges joining two sites in $\mathscr{R}$ are deleted.*
(b) *Only edges crossing l and joining a site in $\mathscr{L}$ to one in $\mathscr{R}$ are created.*
(c) *The worst-case running time of the merge is bounded by a function linear in the sum of three components:*
  (i) *The number of sites examined to find the endpoints of the lower of the two edges of the convex hull of $\mathscr{L} \cup \mathscr{R}$ which cross l.*
  (ii) *The number of edges deleted.*
  (iii) *The number of edges created.*
(d) *The worst-case running time of the merge is $O(|\mathscr{L} \cup \mathscr{R}|)$.*

PROOF. Guibas and Stolfi. □

Our analysis requires some refinement of Theorem 2.6(c). Theorem 2.7 will be used to bound running time analytically, while Corollary 2.8 justifies the use of an easily collected statistic as a measure of efficiency.

THEOREM 2.7. *The running time of the Guibas–Stolfi merge procedure is bounded by a linear function of the number of sites to which it attaches new edges.*

PROOF. By Theorem 2.6(c), it suffices to show that, if new edges are joined to m sites, then at most $3m-6$ edges are created, at most $3m-9$ edges are deleted, and at most m sites are examined to construct the lower convex-hull edge.

Since the edges created are nonintersecting, by Corollary 2.2 there are at most $3m-6$ of them.

Now suppose that $\mathcal{L}$, $\mathcal{R}$, and $\mathcal{L} \cup \mathcal{R}$ consist of n_1, n_2, and n sites, respectively, of which k_1, k_2, and k lie on the convex hull. If c edges are created and d deleted, we have by Lemma 2.1 that

$$(3n_1-k_1-3)+(3n_2-k_2-3)+c-d=3n-k-3.$$

Since $n=n_1+n_2$ and $k \le k_1+k_2$, it follows that $d \le c-3 \le 3m-9$.

Finally, we observe that each site examined to construct the convex-hull edge receives a new edge during the merge. Since each one falls on the boundary of the convex hull of $\mathcal{L}$ or $\mathcal{R}$, it must be the vertex of some angle large than 180° before the merge. However, it must also lie in the interior of the convex hull of $\mathcal{L} \cup \mathcal{R}$, and therefore it cannot be the vertex of such a large angle after the merge. We conclude that no more than m sites are examined. □

COROLLARY 2.8. *The running time of the merge procedure is bounded by a linear function of the number of edges it creates.*

3. The Modified Algorithm and Analysis of Its Worst-Case Running Time.

The unit square is partitioned into $\lceil\sqrt{n/\log n}\rceil^2 = O(n/\log n)$ square cells with sides of length $\lceil\sqrt{n/\log n}\rceil^{-1} \le \sqrt{\log n/n}$. The Delaunay triangulation of the sites within each cell is constructed with the Guibas-Stolfi algorithm. The triangulations within each row of cells are merged in pairs until the triangulation of the row has been completed. Then row triangulations are merged in pairs to complete the triangulation of the entire set of sites (Figure 2.)

```
{ Step 0: Sort Sites into Buckets }
$m := \lceil\sqrt{n/\log n}\rceil$
for $P \in \mathcal{S}$ do insert $P$ into $B_{\lfloor m x_P\rfloor, \lfloor m y_P\rfloor}$
{ Step 1: Triangulate Cells }
for $i := 0$ to $m-1$ do
  for $j := 0$ to $m-1$ do
    $\mathrm{DT}_{ij} :=$ Guibas_Stolfi_DT$(B_{ij})$
{ Step 2: Merge Cells into Rows }
for $k := 0$ to $\lceil\lg m\rceil - 1$ do
  for $i := 0$ to $m-1$ do
    for $j := 0$ to $m-1$ by $2^{k+1}$ do
      $\mathrm{DT}_{ij} := merge(\mathrm{DT}_{ij}, \mathrm{DT}_{i,j+2^k})$;
{ Step 3: Merge Rows }
for $k := 0$ to $\lceil\lg m\rceil - 1$ do
  for $i := 0$ to $m-1$ by $2^{k+1}$ do
    $\mathrm{DT}_{i0} := merge(\mathrm{DT}_{i0}, \mathrm{DT}_{i+2^k,0})$;
return $\mathrm{DT}_{00}$;
```

Fig. 2. Algorithm A.

THEOREM 3.1. *Algorithm A uses $O(n \log n)$ time in the worst case.*

PROOF. Step 0 requires $O(n)$ time, since the floor function is assumed to be computable in constant time.

Number the cells arbitrarily and let n_i be the number of sites in the ith cell. Since Guibas and Stolfi's algorithm requires $O(n \log n)$ time in the worst case, step 1 can be completed in time $\sum_i O(n_i \log n_i) \leq \sum_i O(n_i \log n) \leq O(n \log n)$.

In step 2 no site is involved in more than a single merge for any fixed value of k. Since by Theorem 2.7 the number of points involved in the merges bounds the running time, no more than $O(n)$ time is required for each iteration of the k loop. The k loop is executed $\lg(\lceil\sqrt{n/\log n}\,\rceil) < \lg n$ times, thus $O(n \log n)$ time is required for step 2.

Step 3 can be handled exactly like step 2. Summing the times for the three steps, we find that $O(n \log n)$ time suffices. □

4. Analysis of Expected Time. We will call a distribution with density function f *quasi-uniform* in a region if, for some strictly positive constants c_1 and c_2, $f(x, y) = 0$ outside the region and $c_1 \leq f(x, y) \leq c_2$ inside the region. By modifying constants in the proofs which follow, it is not difficult to show that the expected running time of Algorithm A is $O(n \log \log n)$ for any quasi-uniform distribution in a rectangle. However, we restrict our attention to the uniform distribution in the unit square for the sake of expository simplicity.

During step 1 we accept the $O(n \log n)$ worst-case performance of Guibas and Stolfi's algorithm to construct Delaunay triangulations within each cell. We need only take care to show that it is unlikely that very many sites fall within a single cell, making worst-case performance for that cell unacceptably large.

We next show that when merging two Delaunay triangulations most sites to which new edges are constructed must lie near the boundaries of the two triangulations. We are then able to bound the expected number of sites near enough to the boundary to receive a new edge. By Theorem 2.7, this essentially bounds the running time.

In the sequel $\mathcal{U}$ denotes the unit square, and $\mathcal{S} = \{P_1, P_2, \ldots, P_n\}$ is a set of n sites chosen independently from the uniform distribution on $\mathcal{U}$.

LEMMA 4.1. *The probability that any fixed cell contains more than $e \log n$ sites is less than $1/n$, where e is the base of the natural logarithms.*

PROOF. The probability that a given site lies in a given cell is equal to the area of the cell, which is less than $\log n/n$. Call this probability p, and let N be the number of sites in the cell. Then N has a binomial distribution, and

$$\begin{aligned}\Pr\{N \geq e \log n\} &= \sum_{k \geq e \log n} \Pr\{N = k\} \\ &\leq \sum_{k \geq e \log n} e^{(k - e \log n)} \Pr\{N = k\}\end{aligned}$$

$$\begin{aligned}&\le e^{-e\log n}\sum_{k\ge 0}\binom{n}{k}(ep)^k(1-p)^{n-k}\\&=n^{-e}(ep+1-p)^n\\&=n^{-e}\exp(n\log(1+p(e-1))).\end{aligned}$$

Since $\log(1+x)<x$ for $x>0$, it follows that $\Pr\{N\ge e\log n\}<n^{-e}\exp(np(e-1))<n^{-e}\exp((\log n)(e-1))=n^{-e}\cdot n^{e-1}=1/n$. □

In passing, we note that the same argument can be used to show that the probability that a fixed cell contains more than $(e+m-1)\log n$ sites is at most $1/n^m$.

THEOREM 4.2. *Step* 1 *of Algorithm A requires* $O(n\log\log n)$ *expected time.*

PROOF. Since there are $O(n/\log n)$ cells, by Lemma 4.1 Pr{some cell has $\ge e\log n$ sites} $\le O(n/\log n)\cdot$ Pr{a fixed cell has $\ge e\log n$ sites} $=O(1/\log n)$. Since even these inputs can be handled in $O(n\log n)$ time, their contribution to the expected running time of step 1 is at most $O((n\log n)\cdot(1/\log n))=O(n)$. For the other cases, at most $n/\log n$ subproblems of size at most $e\log n$ must be solved using the $O(n\log n)$ worst-case algorithm of Guibas and Stolfi, making the total expected time $O((n/\log n)e\log n\log(e\log n))$ or $O(n\log\log n)$ for step 1. □

We now bound the expected time of the merging steps. We first show that most new edges lie in a region near the boundaries of the triangulations being merged. Informally, this holds because an edge passing from a site far from the dividing line of the merge across the dividing line to another site must necessarily be long, and so by Corollary 2.4 it must also be a chord of a large site-free circle. Since large regions are unlikely to be site-free, such long edges are also unlikely.

An exception to this observation is a long edge lying entirely near the boundary of the merged triangulation, e.g., an edge of the convex hull. Such an edge may correspond to a large site-free circle which lies mostly outside the region of the triangulation. The edge is not demonstrably improbable, since the portion of the circle lying inside the region of the triangulation may be very small. It is this case which compels us to draw the hyperbolic boundaries in the proof of Theorem 4.5.

Once we have delimited where the likely endpoints of new edges lie, it is an easy matter to calculate the expected number of sites in these regions. A bound on the average running time then follows from Theorem 2.7.

The following lemma will be proved in a more general setting in the next section:

LEMMA 4.3. *Let P, Q, and P′ be three points lying along a line in that order. Let M and N be the other vertices of the square with diagonal PQ. Then any circle passing through P and P′ completely encloses either* $\triangle PQM$ *or* $\triangle PQN$ *or both.*

The next lemma formalizes the assertion that large regions are unlikely to be site-free:

LEMMA 4.4. *Let $p = 2 \log\log n/n$, let $\mathcal{T}$ be a subset of $\mathcal{U}$ with area exceeding p, and let $\mathcal{S}'$ be a set of at least $n-2$ sites drawn independently from the uniform distribution on $\mathcal{U}$. Then* $\Pr\{\mathcal{S}' \cap \mathcal{T} = \emptyset\} = O(1/\log n)$.

PROOF.

$$\begin{aligned}\Pr\{\mathcal{S}' \cap \mathcal{T} = \emptyset\} &\le (1-p)^{n-2} < \exp(-p(n-2)) \\ &\le \log n^{(4/n)-2} = O(1/\log n). \qquad \square\end{aligned}$$

We can now describe the likely location of new edges.

THEOREM 4.5. *Let $\square ABCD$ and $\square CDEF$ lie inside $\mathcal{U}$ and have disjoint interiors, let $\mathcal{R} = \mathcal{S} \cap \square ABCD$ and $\mathcal{L} = \mathcal{S} \cap \square CDEF$, and let $h = |CD|$. Then there is a merge region $\mathcal{M} \subset \square ABCD$ such that:*

(a) $\text{Area}(\mathcal{M}) = O(h\sqrt{\log\log n/n} + \log n \log\log n/n)$,
(b) *if $P \in \mathcal{R}$ but $P \notin \mathcal{M}$ then* $p = \Pr\{\exists P' \in \mathcal{L} \mid (P, P') \in \mathrm{DT}(\mathcal{L} \cup \mathcal{R})\} = O(1/\log n)$.

PROOF. For convenience, we use a coordinate system with origin at C and axes CB and CD (Figure 3). We now verify that conditions (a) and (b) are satisfied by

$$\begin{aligned}\mathcal{M} = \square ABCD \cap \{(x, y) \mid (x \le 8\sqrt{\log\log n/n}) \vee (xy \le 24 \log\log n/n) \\ \vee (x(h-y) \le 24 \log\log n/n)\}.\end{aligned}$$

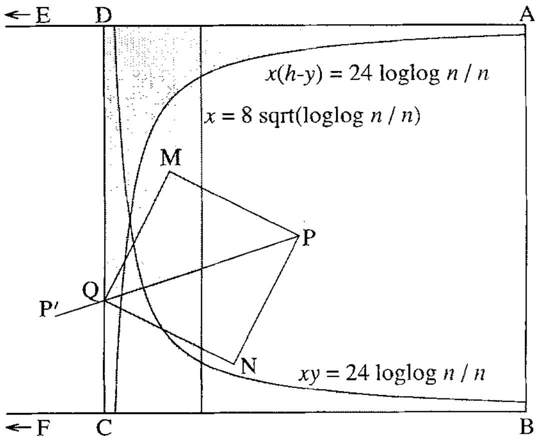

Fig. 3. The merge region $\mathcal{M}$ and $\square PMQN$.

Since the two hyperbolas defining $\mathcal{M}$ are mirror images, and since $|BC| \le 1$, we have

$$\begin{aligned}\mathrm{Area}(\mathcal{M}) &\le 8h\sqrt{\log\log n/n} + 2\int_{8\sqrt{\log\log n/n}}^{1} (24 \log\log n/nx)\,dx \\ &= O(h\sqrt{\log\log n/n} + \log n \log\log n/n).\end{aligned}$$

Now let $P = (x_P, y_P)$ be a site in $\mathcal{R}$ but outside $\mathcal{M}$, let P' be a site in $\mathcal{L}$, and let Q be the point of intersection of PP' and CD. Let $\square PMQN$ be the square with M above and N below PQ. Further suppose that P and P' are joined in $\mathrm{DT}(\mathcal{L} \cup \mathcal{R})$. By Corollary 2.4, there is a circle $\mathcal{C}$ through P and P' whose intersection with $\square ABCD$ is site-free. Then according to Lemma 4.3 either $(\triangle PQM \cap \square ABCD)$ or $(\triangle PQN \cap \square ABCD)$ is site-free. We will show that each of $\triangle PQM$ and $\triangle PQN$ completely contains one of ten regions in $\square ABCD$. Thus the site-free circle $\mathcal{C}$ must also contain one of the regions. But each of the regions has area exceeding $2 \log\log n/n$, so by Lemma 4.4 with $\mathcal{S}' = \mathcal{S} - \{P, P'\}$, the probability that at least one of the regions is site-free is $O(10 \cdot 1/\log n) = O(1/\log n)$—which also bounds the probability that P is joined to a P' in $\mathcal{L}$.

The ten regions are sectors or parts of sectors of the circle of radius $\frac{1}{2}x_P$ centered at P (Figure 4). The central angle of each sector is $\pi/8$. Eight of the regions fall within the sector defined by $\angle CPD$. They must completely cover this sector, but they may otherwise be fixed arbitrarily. To these are added the sector whose upper boundary is PC and the sector whose lower boundary is PD. Since $\angle P'PM$ and $\angle P'PN$ measure $\pi/4$, and $|PM|$, $|PQ|$, and $|PN|$ exceed $\frac{1}{2}x_P$, each of $\triangle PQM$ and $\triangle PQN$ completely contains at least one of these sectors, no matter where P' lies in $\square CDEF$.

The area of each sector is $\frac{1}{2}(\frac{1}{2}x_P)^2(\pi/8) = (\pi/64)x_P^2 \ge \pi \log\log n/n \ge 2 \log\log n/n$. But part of the sector below PC may lie outside $\square ABCD$. In this case let J be the point on PC at distance $\frac{1}{2}x_P$ from P, and let K be the point of intersection of the x-axis and the other side of the sector. Surely the intersection

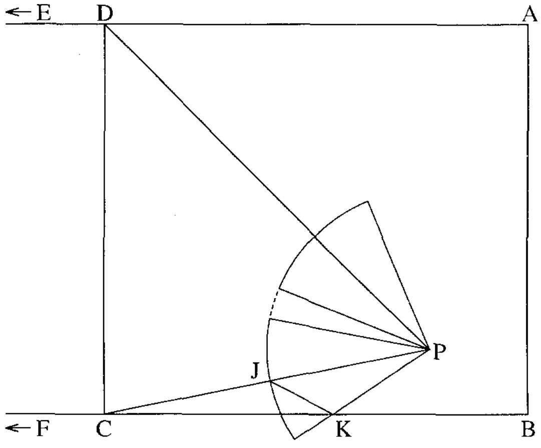

Fig. 4. The ten sectors.

of $\square ABCD$ and the sector contains $\triangle PJK$. But $\text{Area}(\triangle PJK) = \frac{1}{2}\cdot|PJ|\cdot|PK|\cdot\sin(\pi/8) \geq \frac{1}{2}\cdot\frac{1}{2}x_P\cdot y_P\cdot\frac{1}{3} \geq \frac{1}{12}x_P y_P \geq 2\log\log n/n$. Thus we have shown that the intersection of $\square ABCD$ and the sector below PC is itself large enough. The proof is completed by a symmetric argument applied to the sector above PD. $\square$

THEOREM 4.6. *Step* 2 *of Algorithm A requires* $O(n\log\log n)$ *expected time.*

PROOF. The merge region for each merge in step 2 has height $h < \sqrt{\log n/n}$, thus the area of each merge region is $O((\sqrt{\log n/n}\cdot\sqrt{\log\log n/n}) + \log n\log\log n/n) = O(\log n\log\log n/n)$. For a particular value of k there are at most $2^{-k+1}\lceil\sqrt{n/\log n}\rceil^2$ disjoint merge regions with total area $O(2^{-k}\log\log n)$. The expected number of sites in these regions is $O(2^{-k}n\log\log n)$. In addition it is expected that at most another $O(n/\log n)$ sites from outside the merge regions will receive new edges. By Theorem 2.7 the expected running time of step 2 is bounded by

$$\sum_{k=0}^{\lg(\sqrt{n/\log n})} (O(2^{-k}n\log\log n) + O(n/\log n)) = O(n\log\log n) + O(n)$$

$$= O(n\log\log n). \qquad \square$$

THEOREM 4.7. *Step* 3 *of Algorithm A requires* $O(n)$ *expected time.*

PROOF. The merge region for each merge in step 3 has height $h = 1$ and area $O(\sqrt{\log\log n/n})$. There are at most $2^{-k+1}\lceil\sqrt{n/\log n}\rceil$ merge regions for each k value for a total area of $O(2^{-k}\sqrt{\log\log n/\log n}) = O(2^{-k})$ containing $O(2^{-k}n)$ sites in the expected case. Reasoning as for step 2, we find the running time in the expected case to be bounded by

$$\sum_{k=0}^{\lg(\sqrt{n/\log n})} (O(2^{-k}n) + O(n/\log n)) = O(n). \qquad \square$$

Adding together the expected time for each step gives the result:

THEOREM 4.8. *Algorithm A constructs* $\text{DT}(\mathcal{S})$ *in* $O(n\log\log n)$ *expected time.*

5. Extension to the L_p Metrics.

In this section we show that the algorithm of Section 3 and its analysis extend to the L_p metrics for $1 < p \leq \infty$. We are able to make only a small improvement for the L_1 case.

The L_p metric is defined by the distance function $d_p(P, Q) = (|x_P - x_Q|^p + |y_P - y_Q|^p)^{1/p}$ for $1 \leq p < \infty$ and $d_\infty(P, Q) = \max(|x_P - x_Q|, |y_P - y_Q|)$ for $p = \infty$. The L_2 metric is the usual Euclidean metric. The L_1 metric is sometimes called the *rectilinear* or *Manhattan metric.* Hwang [4], Lee and Wong [8], and Lee [6] present $O(n\log n)$ divide-and-conquer algorithms for constructing the Voronoi diagrams in the L_1, L_1 and L_∞, and general L_p metrics, respectively. In these metrics each Voronoi polygon is star-shaped with a nucleus at the associated site, but it is not necessarily convex. The straight-line dual of the Voronoi diagram is called the Delaunay triangulation and denoted $\mathrm{DT}_p(\mathcal{S})$, although in fact $\mathrm{DT}_1(\mathcal{S})$ and $\mathrm{DT}_\infty(\mathcal{S})$ do not generally triangulate the convex hull of $\mathcal{S}$.

Lee [6] shows that Lemma 2.3, Corollaries 2.4 and 2.5, and (essentially) Theorem 2.6 hold for $1<p<\infty$ when the word "circle" is taken to mean the locus of points at a constant distance in the L_p metric from some given point. From these, Theorems 2.7 and 3.1 follow. He also proves the following useful lemma.

LEMMA 5.1. *The locus of points equidistant from two given points, called their* bisector, *is either nonincreasing or nondecreasing if viewed as the graph of a function* $y=f(x)$.

We can now extend the average-case analysis to $1<p<\infty$.

THEOREM 5.2. *Algorithm A constructs* $\mathrm{DT}_p(\mathcal{S})$ *in* $O(n \log\log n)$ *expected time for* $1<p<\infty$.

PROOF. The analysis of Section 4 lacks only an L_p version of Lemma 4.3, which we now provide:

LEMMA 5.3. *Let P, Q, and P' be three points lying along a line in that order. Let M and N be the other vertices of the square with diagonal PQ. Then for* $1 \le p \le \infty$ *any L_p circle passing through P and P' completely encloses either $\triangle PQM$ or $\triangle PQN$ or both.*

PROOF. Let M' and N' be the other vertices of the square with diagonal PP' lying on the same side of PP' as M and N, respectively. Since $\triangle PP'M'$ and $\triangle PP'N'$ contain $\triangle PQM$ and $\triangle PQN$, respectively, it suffices to show that the circle contains either M' or N' (Figure 5). Without loss of generality we consider only the case in which $a \ge b \ge 0$, $P=(0,2b)$, $P'=(2a,0)$, and $M'=(a+b,a+b)$,

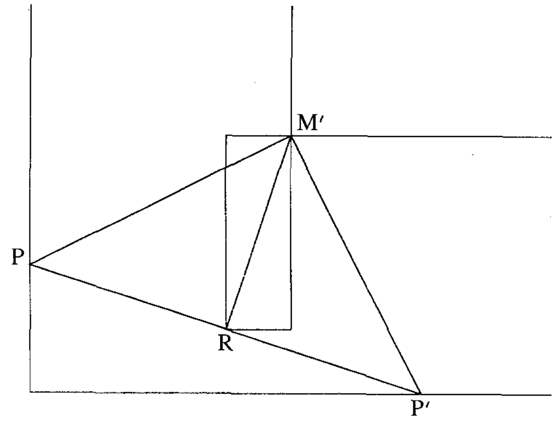

Fig. 5. Bounding the area of L_p circles.

and show that the center of the circle C must lie nearer M' than P if it lies above the line through P and P'.

The triangle inequality implies that the bisector of P and P' passes no nearer to P than $(a^p+b^p)^{1/p}$ at $R=(a, b)$. If $C=R$, M' clearly lies on the perimeter of the circle. Since both M' and R lie on the bisector, it is nondecreasing rather than nonincreasing. Thus $a \le x_C \le a+b$ and $b \le y_C \le a+b$ if C lies on the bisector between R and M', and $d_p(C, M') = (|a+b-x_C|^p + |a+b-y_C|^p)^{1/p} \le (b^p+a^p)^{1/p} = d_p(R, P) \le d_p(C, P)$. Beyond M', C must satisfy $x_C \ge a+b$ and $y_C \ge a+b$, so $d_p(C, M') = ((x_C-(a+b))^p + (y_C-(a+b))^p)^{1/p} \le (x_C^p + (y_C-2b)^p)^{1/p} = d_p(C, P)$. □

It is the L_1 and L_∞ Voronoi diagrams which are of most interest in applications. Unfortunately, implementation and analysis of these cases are complicated by the fact that some convex-hull edges may be omitted from the Delaunay triangulation. For these cases, the Guibas–Stolfi merge procedure must be modified along the lines of Lee's dual algorithm to search downward for the endpoints of the lower convex-hull edge then back upward for the endpoints of the lowest Delaunay edge crossing the dividing line. Thus some sites examined in the search do not receive new edges during the merge, and Theorem 2.7 no longer holds. The analysis of Section 4 still bounds the expected number of edges created, but not the overall running time.

We call the edges of the infinite face of the Delaunay triangulation *boundary edges* and their endpoints *boundary sites*. We show that the L_∞ boundary sites of a set of sites contained in an orthogonal rectangle in the unit square all probably lie near the boundary of the rectangle. There are therefore so few of them in the expected case that searching through them does not dominate the cost of the merge step.

LEMMA 5.4. *A site $P=(x_P, y_P)$ is a boundary site in* $\mathrm{DT}_\infty(\mathcal{S})$ *only if at least one of the quarter-planes defined by the lines $x=x_P$ and $y=y_P$ is site-free.*

PROOF. The result follows immediately from Lee's observation that an edge is a boundary edge in $\mathrm{DT}_\infty(\mathcal{S})$ if and only if one of the two quarter-planes defined by horizontal and vertical rays passing though the endpoints is site-free. □

THEOREM 5.5. *Let $\square ABCD$ be an orthogonal rectangle lying inside $\mathcal{U}$. Then there is a boundary region $\mathcal{B} \subset \square ABCD$ such that*:

(a) $\mathrm{Area}(\mathcal{B}) = O(\log n \log\log n/n)$,
(b) *if $P \in (\mathcal{S} \cap \square ABCD - \mathcal{B})$ then $p = \Pr\{P$ is a boundary site in $\mathrm{DT}_\infty(\mathcal{S})\} = O(1/\log n)$.*

PROOF. For convenience, we use a coordinate system with origin at C and axes CB and CD. We now verify that conditions (a) and (b) are satisfied by

$$\mathcal{B} = \square ABCD \cap \{(x, y) \mid (xy \le 2 \log\log n/n) \vee (x(h-y) \le 2 \log\log n/n) \vee ((w-x)y \le 2 \log\log n/n) \vee ((w-x)(h-y) \le 2 \log\log n/n)\},$$

where $h = |CD|$ and $w = |CB|$.

The bound on Area($\mathscr{B}$) is straightforward since $h, w \leq 1$.

Without loss of generality suppose that $P = (x_P, y_P)$ lies in the lower left quarter of $\square ABCD$ but outside $\mathscr{B}$. Then the smallest of the four rectangles into which $\square ABCD$ is partitioned by the vertical and horizontal lines through P has area $x_P y_P \geq 2 \log\log n/n$. By Lemma 4.4 the probability of its being site-free is at most $O(1/\log n)$. Therefore by Lemma 5.4 the probability that P is on the boundary is at most $O(4/\log n) = O(1/\log n)$. □

We can now bound the running time of Algorithm A for the L_∞ metric.

THEOREM 5.6. *Algorithm A constructs* $\mathrm{DT}_\infty(\mathscr{S})$ *in* $O(n \log\log n)$ *expected time.*

PROOF. The analysis of step 1 in Theorem 4.2 remains valid, since the worse-case performance of the L_1/L_∞ merge procedure is $O(n \log n)$. Since the boundary region of any rectangle is smaller than its merge region, the calculations of Theorems 4.6 and 4.7 for steps 2 and 3 apply to the number of sites examined in boundary-edge searches as well as to the number of sites receiving new edges. Therefore the overall bound of Theorem 4.8 applies as well. □

The L_1 case is more difficult. It follows from Lemma 5.4 and the well-known correspondence between the L_1 and L_∞ metrics that a site P is an L_1 boundary site if and only if one of the quarter-planes defined by the lines through P with slopes $+1$ and -1 is site-free. The L_1 analog of Theorem 5.5 states that all points within a (Euclidean) distance of $\sqrt{2 \log\log n/n}$ of the boundary fall within the boundary region. Analyzing the number of sites examined in lower-boundary-edge searches in steps 2 and 3 yields an $O(n\sqrt{\log n \log\log n})$ bound, which is only a marginal improvement to the original $\Theta(n \log n)$ expected running time.

6. Experimental Results.

A variation of Algorithm A for the Euclidean metric has been implemented for practical evaluation. In this variation the set of n sites is divided into $\lceil\sqrt{n/\log n}\rceil$ equal subsets by horizontal lines. Then the Guibas–Stolfi algorithm, which divides with vertical lines, is applied to each subset. Finally the results are merged in pairs as in step 3 of Algorithm A. (It is notable that the Guibas–Stolfi merge procedure performs the horizontal merges of the second stage without recoding if the sites are sorted by *decreasing* y-coordinate.) This variation is somewhat easier to implement but more difficult to analyze. Intuitively, we expect its behavior to differ little from that of Algorithm A for reasonably large n.

This variation and the original Guibas–Stolfi algorithm were run on inputs generated by drawing sites from the uniform distribution in the unit square. Twenty inputs of size 2^k were generated for $4 \leq k \leq 16$. The results are summarized in Figure 6, which plots the mean number of edges created per site as a function of n, the number of sites. The small variance can be safely ignored. Our measurements for the original algorithm match closely those of Ohya *et al.* It is clear that the modified algorithm is significantly faster for all but the smallest values of n.

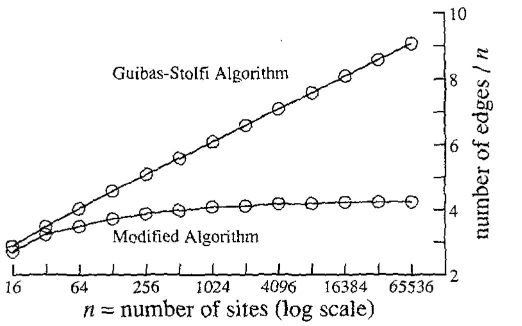

Fig. 6. The algorithms compared.

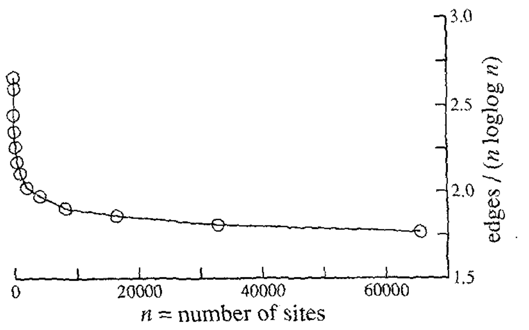

Fig. 7. Estimating the constant factor.

Figure 7 is useful in estimating the constant factor c in the upper bound $cn \log\log n$ on expected running time. If c is asymptotically less than 1.77 as Figure 7 suggests, the number of edges created by the algorithm would be less than $6n$ for $n \leq \exp(6.0/1.77)) \approx 7 \times 10^{12}$. Since about $3n$ edges are required in the final diagram, we conjecture that the running time is no more than twice optimal for n in this range.

7. Conclusions.

We have showed how the average running time of the Guibas-Stolfi algorithm for constructing the Delaunay triangulation can be improved to $O(n \log\log n)$ for a large class of distributions of the sites. We believe the improved algorithm is competitive with existing linear-time algorithms even for for relatively large problems, but cannot state this unequivocably without further theoretical and experimental study of the algorithm's performance on nonuniform distributions and direct experimental comparison with the other algorithms.

We have also showed that the improvement extends to the L_p version of the algorithm for $1 < p \leq \infty$. It would naturally be desirable to extend it to the L_1 case.

Acknowledgments. It is my pleasure to thank Jon Webb, whose code for the Guibas-Stolfi algorithm was the starting-point of this work; Steve Shreve, who provided help with Lemma 4.1; and Ravi Kannan, Cathy McGeoch, and Danny

Sleator, who provided encouragement, engaged in useful discussions, and carefully read and commented upon drafts. Naturally remaining errors are my own.

References

[1] J. L. Bentley, B. W. Weide, and A. C. Yao, Optimal expected-time algorithms for closest point problems, *ACM Trans. Math. Software*, **6** (1980), 563-580.

[2] S. Fortune, A sweepline algorithm for Voronoi diagrams, *Proceedings of the Second Annual Symposium on Computational Geometry*, Yorktown Heights, NY, 1986, pp. 313-322.

[3] L. J. Guibas and J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, *ACM Trans. Graphics*, **4** (1985), 74-123.

[4] F. K. Hwang, An $O(n \log n)$ algorithm for rectilinear minimal spanning trees, *J. Assoc. Comput. Mach.*, **26** (1979), 177-182.

[5] C. L. Lawson, Software for C^1 surface interpolation, in *Mathematical Software III* (J. R. Rice, ed.), Academic Press, New York, 1977, pp. 161-194.

[6] D. T. Lee, Two-dimensional Voronoi diagrams in the L_p-metric, *J. Assoc. Comput. Mach.*, **27** (1980), 604-618.

[7] D. T. Lee and B. Schachter, Two algorithms for constructing Delaunay triangulations, *Internat. J. Comput. Inform. Sci.*, **9** (1980), 219-242.

[8] D. T. Lee and C. K. Wong, Voronoi diagrams in L_1 (L_∞) metrics with two-dimensional storage applications, *SIAM J. Comput.*, **9**, (1980), 200-211.

[9] D. H. McLain, Two-dimensional interpolation from random data, *Comput. J.*, **19** (1976), 178-181 and 384.

[10] A. Maus, Delaunay triangulation and the convex hull of n points in expected linear time. *BIT*, **24** (1984), 151-163.

[11] T. Ohya, M. Iri, and K. Murota, Improvements of the incremental methods for the Voronoi diagram with computational comparison of various algorithms, *J. Oper. Res. Soc. Japan*, **27** (1984), 306-336.

[12] M. I. Shamos and D. Hoey, Closest-point problems, *Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science*, Berkeley, CA, 1976, pp. 208-215.

[13] M. I. Shamos and F. P. Preparata, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[14] R. Sibson, Locally equiangular triangulations, *Comput. J.*, **21** (1978), 243-245.

[15] R. Sibson and P. J. Green, Computing Dirichlet tessellations in the plane, *Comput. J.*, **21** (1978), 168-173.