

There Is a Planar Graph Almost as Good as the Complete Graph

L. Paul Chew
Department of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

Abstract

Given a set S of points in the plane, there is a triangulation of S such that a path found within this triangulation has length bounded by a constant times the straight-line distance between the endpoints of the path. Specifically, for any two points a and b of S there is a path along edges of the triangulation with length less than $\sqrt{10}$ times $|ab|$, where $|ab|$ is the straight-line Euclidean distance between a and b . Thus, a shortest path in this planar graph is less than about 3 times longer than the corresponding straight-line distance. The triangulation that has this property is the L_1 metric Delaunay triangulation for the set S . This result can be applied to motion planning in the plane. Given a source, a destination, and a set of polygonal obstacles of size n , an $O(n)$ size data structure can be used to find a reasonable approximation to the shortest path between the source and the destination in $O(n \log n)$ time.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

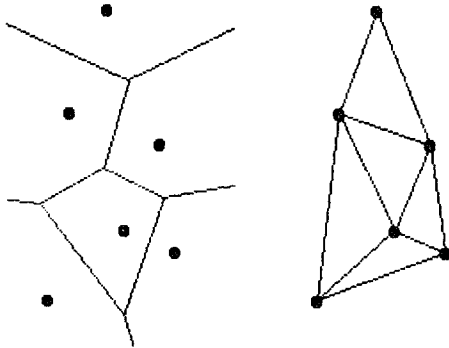
© 1986 ACM 0-89791-194-6/86/0600/0169 \$00.75

Introduction

Let S be a set of n points in the plane. One way to design a network on S in which transmission distances are small is to use the complete graph, the graph with an edge connecting each pair of points in S . The advantage of using this complete network is that the transmission distance between any two points of S is as small as possible. In this paper we show that there is a planar network on S in which transmission distances are at most $\sqrt{10}$ times the optimal distance (the distance in the complete network). Because this network is planar it has $O(n)$ edges instead of the $O(n^2)$ edges needed for the complete network. The planar network that has these properties is the L_1 metric Delaunay triangulation of the set S .

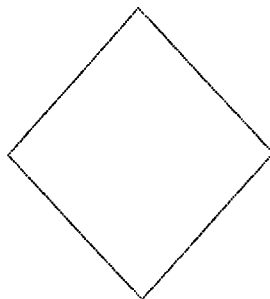
The Delaunay triangulation for a set S of points is most easily described by reference to the Voronoi diagram for the set S . The Voronoi diagram for S divides the plane into regions, one region for each point in S , such that for each region A and corresponding point p , every point within A is closer to p than to any other point of S . The boundaries of these regions form a planar graph. The Delaunay triangulation of S is the straight-line dual of the Voronoi diagram for S ; that is, we connect a pair of points in S if they share a Voronoi boundary. (A more formal definition appears at the end of this section.)

The Voronoi diagram and its dual, the Delaunay triangulation, have been found to be among the most useful data structures in computational geometry. ([LP84] is a survey paper that includes a number of Voronoi diagram applications.)



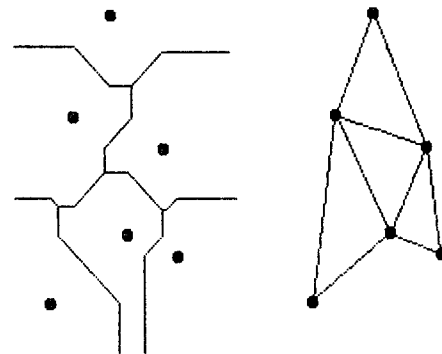
A Voronoi diagram and the corresponding Delaunay triangulation.

The results presented in this paper use a Delaunay triangulation based on the L_1 metric instead of the standard Euclidean metric. For the Euclidean (L_2) metric, distances are calculated using the familiar distance formula (the square root of the x -distance squared plus the y -distance squared). For the L_1 metric, the distance between two points is defined as the absolute value of the x -distance plus the absolute value of the y -distance. Just as in the Euclidean metric, a circle is defined as the set of points equidistant from a chosen center point. Thus, a circle in the L_1 metric is diamond shaped, i.e., a square tipped at 45° .



An L_1 circle.

The L_1 Voronoi diagram is defined just like the standard (Euclidean) Voronoi diagram except the L_1 metric is used to calculate distances. Like the standard Voronoi diagram, the boundaries of the L_1 Voronoi regions form a planar graph. Both the Euclidean Voronoi diagram and the L_1 Voronoi diagram (and Voronoi diagrams based on many other distance functions) can be constructed in $O(n \log n)$ time where n is the number of points in the set S (see, for instance, [SH75, Hw79, CD85]). The L_1 Delaunay triangulation can be derived from the corresponding Voronoi diagram in $O(n)$ time, or, alternately, it can be built directly using a method similar to the Euclidean-case method presented in [LS80].



An L_1 Voronoi diagram and the corresponding L_1 Delaunay triangulation.

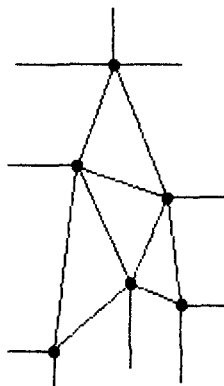
A Delaunay triangulation has the property that, for any empty triangle of the triangulation, the circumscribed circle contains no points of S in its interior. Of course, for the L_1 Delaunay triangulation, the circle in question is the tipped square described above. We define a Delaunay triangulation in terms of empty circles.

Definition. A *Delaunay triangulation* of S is a triangulation T of S such that for each empty triangle of T

- 1) there is a circle that can be circumscribed about the triangle, and
- 2) the interior of this circle contains no points of S .

If a set S contains 4 or more points that lie on a single empty circle then S has more than one possible Delaunay triangulation. For the results presented in this paper any of these Delaunay triangulations may be used. Part 1 of the definition is unnecessary for the standard (Euclidean) Delaunay triangulation since there is a circumscribed circle for any triangle. For the L_1 metric, however, there exist triangles that do not have circumscribed circles (consider, for example, two points on a horizontal line with a third point between them and just above the line).

To simplify our presentation we assume S contains the 4 points $(\pm\infty, 0)$ and $(0, \pm\infty)$. Thus a triangulation of S is always a triangulation of the entire plane. This assumption simplifies some of the definitions and avoids a number of special cases in proofs. Of course these points at infinity are never used on a shortest path so applications of the results presented here do not require these extra infinities.



An L_1 Delaunay triangulation using points at infinity.

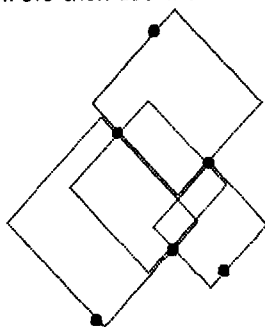
The remainder of this paper includes an outline of the proof that paths in the L_1 Delaunay triangulation are of bounded length, a discussion of how to apply this result to motion planning in the plane, and some directions for further research. Of particular interest is the possibility of applying techniques similar to those presented here to create an $O(n^2)$ algorithm for almost optimal motion planning in 3 dimensions.

The Proof

The proof is in three parts. First, we consider a special case. We assume that S contains points a and b such that the line between a and b is horizontal. Using this assumption, we show there is a path from a to b in the circle graph (defined below) of length $\leq 2\sqrt{2}|ab|$, where $|ab|$ is the Euclidean distance from a to b . In the second part we remove the restriction that segment ab is horizontal; we show that for any points a and b of S there is a path in the circle graph of length $\leq \sqrt{10}|ab|$. Finally, we complete the proof by showing there exists a path from a to b using edges of the Delaunay triangulation with length \leq the length of the path in the circle graph. We start with the definition of a circle graph.

Definition. Let T be a Delaunay triangulation of a set S of points in the plane. The vertices of the *circle graph* derived from T are the points of S . The *circle graph* has three edges for each empty triangle Δ of T . These edges correspond to the three arcs of an empty circle circumscribed about Δ . Further, the length of an edge is defined to be the length of the corresponding arc. If there is more than one empty circle that can be circumscribed about a triangle then use the

smallest such circle; if there is more than one smallest circle then use the leftmost such circle.



An L_1 circle graph (the infinite circles are not shown).

Note that for each edge e in T there are two edges in the circle graph, one derived from each of the triangles that include e . Of course, the circle graph uses circles based on the appropriate distance function. Thus, for us, the circles in the circle graph are tipped squares.

Lemma 1. Let T be an L_1 Delaunay triangulation of a set S of points in the plane and let G be the circle graph derived from T . Assume S contains points a and b such that the line between a and b is horizontal. There is a path from a to b in G such that the length of the path is $\leq 2\sqrt{2}|ab|$, where $|ab|$ is the Euclidean distance from a to b .

We prove the existence of the desired path by giving an algorithm to compute it.

0. Let L be the horizontal line segment from a to b with a at the left. Without loss of generality we may assume that no vertices of T except a and b lie on L (if there were such a vertex, say c , we could recursively find paths from a to c and from c to b and put them together to make a path from a to b). Let T' include just the

vertices and edges of the Delaunay triangles that have a part of L in their interior. The remainder of the algorithm uses only vertices and edges of T' . Note that the triangles of T' are ordered from left to right along L . We use x to represent our current position on the path as it is constructed. Initially x is the point a .

1. Let Δ be the rightmost of all triangles of T' that have x as a vertex and let C be the empty circle (square) corresponding to Δ in the circle graph. Note that, by choosing the rightmost triangle, one of the remaining vertices of Δ (call it x_1) is above L and clockwise along C from x and the other vertex of Δ (call it x_2) is below L and counterclockwise along C from x . (It's also possible that one of the vertices is b and is thus on L . We consider the point b to be both above and below L .)
 2. **if** x is on the upper left edge of C
 then move clockwise around C
 else if x is on the lower left edge of C
 then move counterclockwise around C
 else if x is above L
 then move clockwise around C
 else if x is below L
 then move counterclockwise around C
 3. Continue in the same direction around C until either x_1 or x_2 is reached. Whichever is reached, call it x . If x is b then quit else go to step 1.

We prove a series of lemmas describing the shape of the path produced by the algorithm. These results are then used to complete the proof of Lemma 1.

Lemma 2. The triangles used (called Δ in step 1) are ordered along L . Although not all triangles of T' are used, those used appear in their order along L .

Proof. Follows from the use of rightmost triangles. \square

Since there are only finitely many triangles it follows from Lemma 2 that the algorithm terminates.

Corollary. The algorithm terminates, producing a path from a to b .

Lemma 3. Let Δ_1 and Δ_2 be triangles of T' and let C_1 and C_2 be the corresponding circles in the circle graph. Let e_1 and e_2 be the rightmost intersections of C_1 and C_2 (respectively) with L . If $\Delta_1 \preceq \Delta_2$ then $e_1 \preceq e_2$. (Here, \preceq is used to represent both the ordering of triangles along L and the ordering of points of L .)

Proof. It is sufficient to prove the lemma for Δ_1 and Δ_2 adjacent along L . The result follows from the fact that each circle contains no vertices of T in its interior. \square

Lemma 4. Let p be a point travelling from a to b along the path constructed by the algorithm given above.

1) If p is above L then it does not move in the direction left-up.

2) If p is above L then p moves in the direction right-up iff it is travelling along the upper left edge of the current circle C .

3) Similar statements hold for p below L .

Proof. These results follow from the choices allowed in step 2 of the algorithm. \square

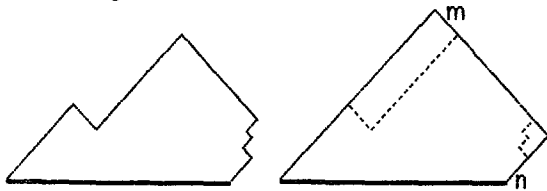
Lemmas 2 through 4 are tools used to determine the possible shapes for the path created by the algorithm.

Proof of Lemma 1. We first divide the path into pieces, then we analyze the shapes of the pieces to prove the lemma. As a first step, the path created by the algorithm is extended so the path hits L more often. Specifically, whenever the path turns to the left (in the direction left-up or left-down) the path is extended to hit L before it is allowed to turn away from L . For example, let p be a point travelling along the path and suppose p is above L and has just turned in the direction left-down. Whether this portion of the path needs to be extended depends on what happens as p continues to move along the path. We extend this portion of the path if and only if p moves away from L before it hits L . At the point where p turns away from L we extend the path, forcing p to move in the direction left-down until L is reached. To continue the path, p moves back from L in the direction right-up until the original path is rejoined. For p below L and moving in the direction left-up, the path is extended in a similar manner. Lemma 4 shows p above L moving left-down and p below L moving left-up are the only cases in which p moves left.



Extending the path when it heads left.

This extended path is chopped into pieces, breaking the path wherever it hits L . Due to constraints on the direction of the path (Lemma 4) and restrictions on how circles are ordered (Lemma 3) we can determine the basic shape of each piece of the extended path. Intuitively, each piece can be "unfolded" without changing the path length to produce a shape similar to the following picture.



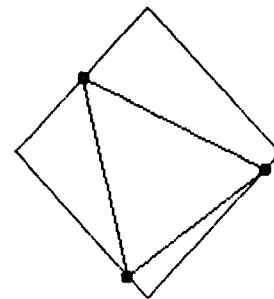
A portion of the path and its equivalent "unfolded" path.

Using Lemma 3, it can be shown that n must be to the right of m (see diagram). It follows that for each piece of the path the ratio of the length of the path to the length of its portion of L is bounded by $2\sqrt{2}$. \square

What happens if the line between a and b is not horizontal? Or, equivalently, what happens if the circles we use are squares tipped at an angle other than 45° ? Let δ be the angle at which the squares are tipped. The ratio of path length to distance covered along L can be shown to be $\sin\delta + 3\cos\delta$ for sections of path above L and $\cos\delta + 3\sin\delta$ for sections of path below L . Some simple calculus shows that this ratio can be at most $\sqrt{10}$, giving the following lemma.

Lemma 5. Let T be an L_1 Delaunay triangulation of a set S of points in the plane and let G be the circle graph derived from T . For any two points a and b in S there is a path from a to b in G such that the length of the path is $\leq \sqrt{10} \cdot |ab|$, where $|ab|$ is the Euclidean distance from a to b .

We improve this result by observing that the path created by the algorithm travels along the boundary of a circumscribed circle to get from vertex to vertex of a triangle. Note that the edge of the triangle is a shortcut between these vertices. In other words, instead of travelling along the edges of the circle graph we can visit the same vertices by travelling along the edges of the Delaunay triangulation; in addition, this path is shorter than the path in the circle graph.



The triangle edges are shortcuts between vertices.

Theorem 1. Let T be an L_1 Delaunay triangulation of a set S . For any two points a and b in S there is a path from a to b in T such that the length of the path is $\leq \sqrt{10} \cdot |ab|$, where $|ab|$ is the Euclidean distance from a to b .

Application to Motion Planning

With some modification, this result can be applied to motion planning in the plane. Let s be a source

point and let d be a destination point. Let R be a set of polygonal obstacles in the plane where n is the number of vertices in R . For simplicity of presentation we assume s and d are included in R as degenerate polygonal obstacles. The motion planning problem is to find the minimum length s -to- d path that does not collide with any of the obstacles in R .

The optimal path can be found by using the visibility graph, a graph in which points are connected iff they can "see" each other. Dijkstra's algorithm can be used to determine the shortest path in the n vertex visibility graph in $O(n^2)$ time. The visibility graph itself can be constructed in $O(n^2)$ time [We85, AAGHI85].

A Delaunay-like graph G (here, called an *obstacle triangulation*) can be built on the vertices of R . G has the following properties:

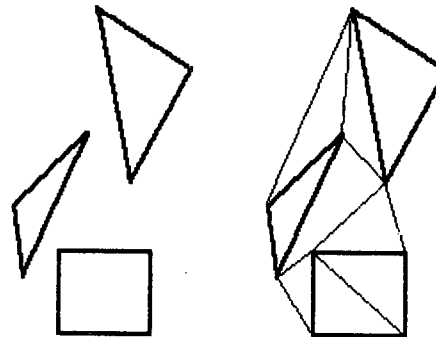
- 1) G is planar;
- 2) the minimum length G -restricted path (i.e., a path restricted to edges of G) from s to d has length at most $\sqrt{10}$ times the length of the optimal path.

Intuitively, the obstacle triangulation is a Delaunay triangulation with the obstacle edges forced in as part of the triangulation. Compare the following definition of an obstacle triangulation with the definition of a Delaunay triangulation given in the introduction.

Definition. An *obstacle triangulation* of R is a triangulation T of R such that all edges of R are included in T , and such that for each edge e of T either e is an edge of R or there exists a circle C with the following properties

- 1) the endpoints of edge e are on C , and
- 2) if any vertex of R is in the interior of C then it

cannot be "seen" from either endpoint of e (i.e., a line segment drawn from such an interior vertex to an endpoint of e must cross an edge of R).



Obstacles and the corresponding obstacle triangulation.

The obstacle triangulation can be built using a straightforward algorithm in $O(n^2)$ worst-case time. Conjecture: the obstacle triangulation can be constructed in $O(n \log n)$ worst-case time.

Because the obstacle triangulation is a planar graph, Dijkstra's algorithm can be used to find the shortest path within the obstacle triangulation in $O(n \log n)$ time. The following result shows that this path is a good approximation to the shortest possible path.

Theorem 2. Let R be a set of polygonal obstacles in the plane and let G be the obstacle triangulation of R . If s and d are vertices of R then the shortest G -restricted (using only edges of G) path from s to d is of length $\leq \sqrt{10}$ times the length of the shortest possible path from s to d .

Proof Let P be the optimal path from s to d . Consider a straight line portion of P say from a to b , where a and b are vertices of R , such that no other vertices of R appear on this portion of the path. With minor modifications, Theorem 1 can be extended to show

there is a path from a to b in G with length $\leq \sqrt{10} \cdot |ab|$. (It is necessary to show that Lemma 3 holds for G .) Since such a path in G exists for each straight-line portion of P , it follows that there is a path in G from s to d with length $\leq \sqrt{10}$ times the length of the optimal path. \square

Further Research

There are several interesting possibilities for extending the results presented here. The constant ($\sqrt{10}$) that bounds the ratio of path lengths is unlikely to be optimal. In other words, the length of a shortest path in the Delaunay triangulation (or the obstacle triangulation) is probably closer to the length of the optimal path than has been shown here. It can be shown that $\sqrt{2}$ is a lower bound for this constant, but this is not a tight bound. The original motivation for this research was to prove results similar to those presented here, but for the standard (Euclidean) Delaunay triangulation. Conjecture: the ratio of path lengths for the Euclidean Delaunay triangulation is bounded by a constant. It can be shown that $\pi/2$ is a lower bound for this constant.

Of particular interest is the possibility of extending these results to motion planning in 3 (or more) dimensions. Currently, the best methods known for finding an optimal path among polyhedral obstacles in 3 dimensions are not even polynomial [SS84, Sh85]. Papadimitriou [Pa85] has developed a fully polynomial approximation algorithm. His method is polynomial in n , the number of elements (vertices, edges, and faces) in the obstacle set, and $1/\delta$ where $(1+\delta)$ is the desired bound on the ratio of approximate path over shortest path. It may be possible to create a graph similar to the obstacle triangulation, with edges connecting vertices of the polyhedrons. The

ratio of a shortest path within this graph to the actual shortest path should be bounded by a constant. The result would be an $O(n^2)$ algorithm for finding a close-to-optimal path among polyhedrons.

References

- [AAGHI85] T. Asano, T. Asano, L. Guibas, J. Hershberger, H. Imai, Visibility-polygon search and Euclidean shortest paths, Proc. 26th IEEE Symposium on Foundations of Computer Science (1985), 155-164.
- [CD85] L. P. Chew and R. L. Drysdale, Voronoi diagrams based on convex distance functions, Proc. 1st Symposium on Computational Geometry, Baltimore (1985), 235-244.
- [Hw79] F. K. Hwang, An $O(n \log n)$ algorithm for rectilinear minimal spanning trees, JACM 26 (1979), 177-182.
- [LS80] D. T. Lee and B. Schachter, Two algorithms for constructing Delaunay triangulations, International Journal of Computer and Information Sciences, 9:3 (1980), 219-242.
- [Pa85] C. H. Papadimitriou, An algorithm for shortest-path motion in three dimensions, Information Processing Letters, 20 (1985), 259-263.
- [SH75] M. I. Shamos and D. Hoey, Closest-point problems, Proc. 16th IEEE Symposium on Foundations of Computer Science (1975), 151-162.

- [Sh85] M. Sharir, On shortest paths amidst convex polyhedra, Tech. Rept. 181, Computer Science Division, Courant Institute of Mathematical Sciences (1985).
- [SS84] M. Sharir and A. Schorr, On shortest paths in polyhedral spaces, Proc. 16th ACM Symposium on Theory of Computing (1984), 144-153.
- [We85] E. Welzl, Constructing the visibility graph for n line segments in $O(n^2)$ time, Information Processing Letters, 20 (1985), 167-171.