Improving shortest paths in the Delaunay triangulation *

Manuel Abellanas† Mercè Claverol‡ Gregorio Hernández† Ferran Hurtado§ Vera Sacristán§
Maria Saumell§ Rodrigo I. Silveira§

Abstract

We study a problem about shortest paths in Delaunay triangulations. Given two nodes s, t in the Delaunay triangulation of a point set P, we look for a new point p that can be added, such that the shortest path from s to t, in the Delaunay triangulation of $P \cup \{p\}$, improves as much as possible. We study several properties of the problem, and give efficient algorithms to find such point when the graph-distance used is Euclidean and for the link-distance. Several other variations of the problem are also discussed.

1 Introduction

There are many applications involving communication networks where the underlying physical network topology is not known, too expensive to compute, or there are reasons to prefer to use a logical network instead. An example of an area where this occurs is ad-hoc networks, where nodes can communicate with each other when their distance is below some threshold. Even though the routing is done locally, to avoid broadcasting to all neighbors every time a packet needs to be sent, some logical network topology and routing algorithm must be used.

Similar situations arise in application-layer routing, where sometimes a logical network is used on top of the actual, physical network (see for example [5]). This logical network is assumed to have some overlay topology, whose choice can have an important impact on the overall performance of the network.

The Delaunay triangulation is often used to model the overlay topology [3, 5] due to several advantages: it provides locality, scales well, and in general avoids high-degree vertices, which can create serious bottlenecks. In addition, several widely-used *localized* routing protocols guarantee to deliver the packets when the underlying network topology is the Delaunay triangulation [2]. Furthermore, there are localized routing protocols based on the Delaunay triangulation where the total distance traveled by any packet is never more than a small constant factor times the network distance between source and destination (e.g. [2]). Since the Delaunay triangulation is known to be a spanner [4], in the case of geometric networks this guarantees that all packets travel at most a constant times the minimum travel time.

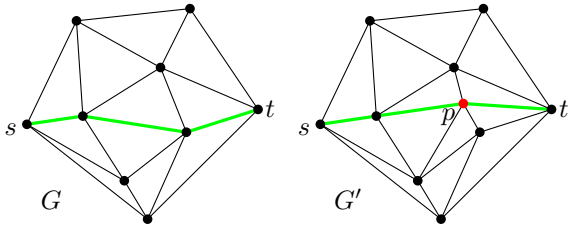

Figure 1: Shortest path between s and t before (left) and after (right) adding p, resulting in a shorter path.

In this paper we consider the problem of improving a geometric network, with a Delaunay triangulation topology, by augmenting it with additional nodes. In particular, we aim at improving the shortest path on the Delaunay network between two given nodes s and t. Adding new nodes to a Delaunay network produces changes in the network topology that can result in equal, shorter, or longer shortest paths between s and t (see Figure 1).

We restrict ourselves to the scenario where at most one node can be added to the network, which can be placed anywhere on the plane. The goal is to find a location for the new node that improves the shortest path between s and t as much as possible. We are not aware of any previous work on this problem.

Notation The input to the problem is a set of n points $P = \{p_1, \cdots, p_n\}$, and two points $s, t \in P$. The points represent the locations of the network nodes.

We will use G to denote the Delaunay graph of P: G has the points in P as vertices, and an edge between two vertices (p_i, p_j) if and only if there is a circle through p_i, p_j that does not contain any point

*This research was initiated during the 6th Iberian Workshop on Computational Geometry, held in Aveiro, Portugal. M.A. and G.H. were partially supported by projects MTM2008-05043 and HP2008-0060. M.C., F.H., V.S. and M.S. were partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040. R.I.S. was supported by the Netherlands Organisation for Scientific Research (NWO).

†Facultad de Informática, Universidad Politécnica de Madrid, {mabellanas, gregorio}@fi.upm.es.

‡Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, merce@ma4.upc.edu

§Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, {ferran.hurtado, vera.sacristan, maria.saumell, rodrigo.silveira}@upc.edu

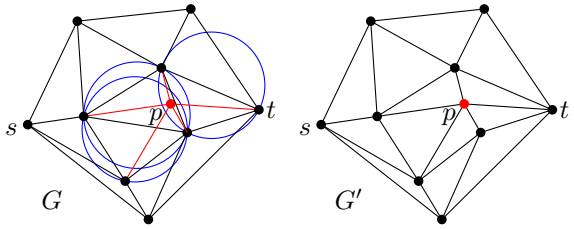

Figure 2: Adding a new point p to the Delaunay triangulation.

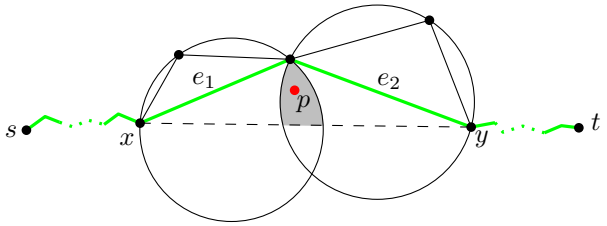

Figure 3: Any point inserted in the shaded region, like p, improves $SP(s,t)$, shown in green.

from P in its interior. We assume the points in P are in general position: no three points are collinear and no four points are cocircular. Thus G represents the Delaunay triangulation of P. Moreover, we also assume that the edge $(s,t) \notin G$, otherwise the distance between s and t in G would be optimal. Note that we use G to refer to both the graph and the triangulation.

The shortest path on G between s and t will be denoted by $SP_G(s,t)$. The length of such path, defined as the sum of the Euclidean lengths of its edges, will be denoted by $|SP_G(s,t)|$, although we will omit G if possible. The straight line segment between two points x and y will be denoted by $\overline{xy}$, and its Euclidean length by $|\overline{xy}|$, whereas the Euclidean length of an edge (x,y) will be denoted by $|(x,y)|$.

Finally, we will use G'_p to denote the Delaunay graph of $P \cup \{p\}$, for some $p \notin P$ (we will omit p when clear from the context).

2 Properties and observations

We begin the paper by analyzing some geometric properties of the problem.

When a new point p is inserted in P, some edges of the Delaunay triangulation might disappear and new edges, all incident to p, appear (see Figure 2). The edges of G that are affected by the insertion of p belong to Delaunay triangles whose circumcircles contain p. In particular, all triangles in G whose circumcircle contains p get new edges in G', connecting their vertices to p. If p is outside the convex hull of P, some additional edges might appear.

A first question that one may ask is whether it is always possible to improve a shortest path by adding one point to G. There are situations in which it is easy to obtain *some* improvement:

Lemma 1 *Let e_1 and e_2 be two consecutive edges of $SP_G(s,t)$. Let C_1 and C_2 be two Delaunay circles through the extremes of e_1 and e_2, respectively. If C_1 and C_2 are not tangent and $C_1 \cap C_2$ is on the side in which the edges form the smallest angle, then the length of $SP_G(s,t)$ can always be reduced by inserting one point.*

Even though we omit the proof of the previous lemma, the basic idea can be seen in Figure 3. If we insert p in the shaded region, then (x,p) and (p,y) will be Delaunay edges in G', shortening the part of $SP(s,t)$ between x and y.

However, some shortest paths cannot be improved at all by adding a single point:

Lemma 2 *It is sometimes impossible to improve $SP_G(s,t)$ by inserting only one point.*

Proof. An example is shown in Figure 4. Notice that e_1, e_2, C_1, and C_2 do not satisfy the hypothesis of Lemma 1. Moreover, $\forall x \in P \setminus \{s,t\}$, $|xs| + |xt| \geq |e_1| + |e_2|$, so a shorter path between s and t must be of the form $\{s,p,t\}$. More precisely, p must be inserted so that $|sp| + |pt| < |e_1| + |e_2|$, $(s,p) \in G'_p$, and $(t,p) \in G'_p$. It is easy to verify that these conditions cannot be simultaneously satisfied. □

The example in Figure 4 can be extended to a path with a linear number of vertices, and even to one where $SP(s,t)$ zigzags.

On the other hand, two points always suffice to improve the shortest path between s and t.

Lemma 3 *It is always possible to reduce the length of $SP_G(s,t)$ by inserting two points, if $SP_G(s,t)$ is not optimal.*

Proof. Let $e_1 = (a,b)$, $e_2 = (b,c)$, C_1 and C_2 be two consecutive edges of $SP(s,t)$, and two Delaunay cir-

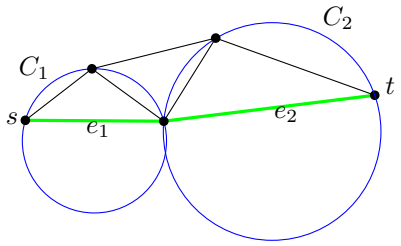

Figure 4: Example where $SP(s,t)$ cannot be improved by adding one point.

cles circumscribing them. If Lemma 1 cannot be applied, it is easy to see that we can place two points p_1 and p_2 close enough to b so that $\{a, p_1, p_2, c\}$ is a path in the new triangulation, shorter than $\{a, b, c\}$. □

3 Finding a point that gives the maximum improvement

In this section we present an algorithm that computes a point p such that $|SP_{G'_p}(s,t)|$ is minimum. The correctness of the algorithm is based on the following lemmas. First we prove that we only need to look at $O(I)$ possible candidate points, where I is the number of pairs of Delaunay circles that intersect—$I \in \Theta(n^2)$ in the worst case. Then we show that the shortest paths from s and t need to be computed only once.

Lemma 4 *Let p be an optimal point, and let x, y be the points in G such that $SP_{G'_p}(s,t)$ includes (x,p) and (p,y) as consecutive edges. Then p lies on the segment $\overline{xy}$, or on the intersection of a circumcircle through x, and a circumcircle through y.*

Proof. Suppose that p does not lie on the straight line segment $\overline{xy}$. Assume w.l.o.g. that $\overline{xy}$ is horizontal, and p is above $\overline{xy}$. Since p is optimal but moving p down reduces $|xp| + |py|$, any movement toward $\overline{xy}$ must result in crossing a circle C that causes a flip removing (x,p), (p,y) or some other edge in $SP_{G'}(s,t)$. This can be because p either enters or leaves C.

If p enters a circle, an edge e connecting two neighbors of p disappears, and is replaced by an edge e' incident to p. Such a change cannot affect the combinatorial structure of the current shortest path $SP_{G'}(s,t)$.

Hence we are only interested in the event of leaving a circle C. In this case an edge e, until now incident to p, disappears, and is replaced by another edge e' (see Figure 5). This can be a problem only if $e = (x,p)$ or $e = (p,y)$. Assume that $e = (x,p)$. Notice that x lies on C. Even though moving p down would make it cross C, the length of $SP_{G'}(s,t)$ could also be reduced by moving p on the circle C toward $\overline{xy}$. If such a movement cannot be done without affecting the combinatorial structure of $SP_{G'}(s,t)$, we conclude that there is a second circle C' through p and y. □

Lemma 5 *Let p be a point, and let $x \in P$ such that $(x,p) \in G'_p$. If $SP_{G'_p}(s,p)$ includes (x,p), then $|SP_{G'_p}(s,p)| = |SP_G(s,x)| + |xp|$. Otherwise, $|SP_{G'_p}(s,p)| \leq |SP_G(s,x)| + |xp|$.*

Algorithm The previous lemmas imply that to find an optimal point p it is enough to analyze each pair of Delaunay circles that intersect.

We first precompute the shortest path trees from s and from t. Then we use an output-sensitive algorithm to compute all pairs of circles that intersect.

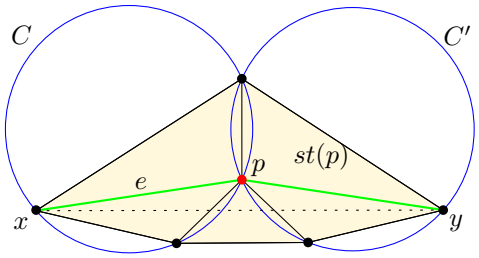

Figure 5: Situation in proof of Lemma 4.

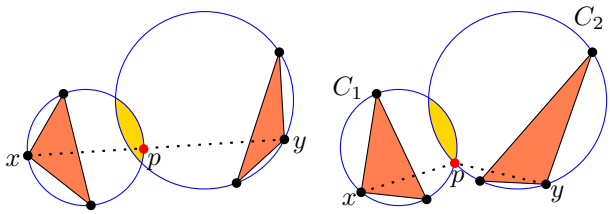

Figure 6: Two optimal ways to connect x and y: by a point on $\overline{xy}$ (left), and by a point on the boundary of $C_1 \cap C_2$ (right).

For each pair of intersecting circles (C_1, C_2), we proceed as follows. Each circle corresponds to a Delaunay triangle from G. Let the two triangles be t_1, t_2. For each pair of vertices $x \in t_1$ and $y \in t_2$, we first check if $\overline{xy}$ intersects $C_1 \cap C_2$. If it does, we take p as any point on $(\overline{xy} \cap C_1 \cap C_2)$. Otherwise, we check the two points where C_1 and C_2 intersect, and use the one that gives the shortest path from x to y. See Figure 6.

If the length of $SP(s,t)$ improves by using p, we update this information. In the end we output the point that gave the shortest path, if it improves over $SP_G(s,t)$, or report that no point can improve it.

The running time of the algorithm is dominated by the time needed to find all pairs of circles that intersect. Therefore by using an algorithm like Balaban's [1] we obtain an $O(n \log n + I)$ running time, with $O(n)$ space.

Theorem 6 *Given the Delaunay triangulation of n points, and two vertices s and t, a point whose insertion gives the maximum improvement in $SP(s,t)$ can be found in $O(n \log n + I)$ time, where I is the number of pairs of Delaunay circles that intersect.*

3.1 Related problems

Based on the previous lemmas, some other related problems can also be solved more efficiently.

Finding an optimal point for every t If only s is fixed, one may want to compute for each $t \in P$ a

point p_t whose insertion gives the optimal improvement in $SP(s,t)$. This can be done efficiently in two steps. First we augment G by adding some edges as follows: we add an edge (x,y) of weight w if there exists a circumcircle through x intersecting a circumcircle through y, such that the best point p in the intersection satisfies $|xp|+|py|=w$. These new edges can be found in $O(n\log n+I)$ time, and the resulting graph H has $O(n+I)$ edges and n vertices. In the second step, we use Dijkstra's algorithm implemented with Fibonacci heaps to compute the single-source shortest paths from s in H, modified to ensure that no path uses more than one of the new edges. This yields a running time of $O(n\log n+I)$. This approach can also be used to compute, for each s and each t, the point p_{st} that gives the optimal improvement for s and t, in $O(n^2\log n+nI)$ time.

Other network graphs If instead of the Delaunay graph a different proximity graph is used (e.g. Gabriel or nearest neighbor graph), one can apply the general approach of partitioning the plane into regions such that inserting a point anywhere inside the region produces the same topological change to the structure. Then the exact optimal location within a region can be computed. Several proximity graphs related to the Delaunay graph can use such technique, including the minimum spanning tree (the corresponding subdivision has been studied in [6]).

4 Finding a point that gives the maximum improvement, using the link-distance

Next we consider the variant of the problem where the metric used to measure distances on G is the link-distance: the length of a path is defined by its number of edges. This metric is also interesting in networking applications, since it measures the number of hops.

We use $d_l(s,t)$ to denote the link-distance between s and t. As before, we are interested in adding one new point to G such that the link-distance between s and t is minimized as much as possible.

Data structure We use a data structure D that allows to answer the following type of query in $O(\log^2 n)$ time: given a circle C, find a Delaunay circle C^* that (i) intersects C, and (ii) has minimum link-distance to t. The link-distance from a circumcircle C—corresponding to a triangle $\triangle(a,b,c)$—to t is defined as $\min\{d_l(a,t),d_l(b,t),d_l(c,t)\}$. The data structure consists of a balanced binary tree where each node represents the union of a subset of the circumcircles. The unions are represented by additively-weighted Voronoi diagrams of the circle centers. It takes $O(n\log n)$ space and can be built in $O(n\log^2 n)$ time. We omit the details due to the space limit.

Algorithm The algorithm proceeds as follows. First the shortest paths from s and t to each other node are precomputed. Then we go over all points in P. For each possible point x we query the data structure D with each circumcircle that goes through x. We simply keep track of the lowest value returned by each query, for each point. After doing this for all points in P, we return a point in the intersection associated with the circles that gave the minimum distance. The correctness of the algorithm follows from the previous lemmas, which also hold for the link-distance. The total running time of the algorithm is $O(n\log^2 n)$.

Theorem 7 *Given the Delaunay triangulation of n points, and two vertices s and t, a point whose insertion gives the maximum improvement in $SP(s,t)$, under the link-distance, can be found in $O(n\log^2 n)$ time.*

5 Discussion

We studied the problem of adding a point to a Delaunay triangulation, such that it improves a shortest path as much as possible. As already mentioned, the methods and observations used can be adapted to solve other related problems, like when other proximity graphs are used. It is also possible to solve efficiently the somewhat dual problem of finding a node whose removal gives the best improvement to the shortest path between s and t. We omit details due to space constraints.

Several improvements to our algorithms are possible. A particularly intriguing question is whether the decision problem (Is there a point that improves $SP(s,t)$?) can be solved faster than the optimization version.

References

[1] I. J. Balaban. An optimal algorithm for finding segments intersections. In *Proc. SoCG'95*, pages 211–219, 1995.

[2] P. Bose and P. Morin. Online routing in triangulations. *SIAM J. Comput.*, 33(4):937–951, 2004.

[3] E. Buyukkaya and M. Abdallah. Efficient triangulation for P2P networked virtual environments. In *Proc. NetGames'08*, pages 34–39, 2008.

[4] D. P. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete Comput. Geom.*, 5:399–407, 1990.

[5] J. Liebeherr, M. Nahas, and W. Si. Application-layer multicasting with Delaunay triangulation overlays. *IEEE J. Sel. Areas Comm.*, 20:1472 – 1488, 2002.

[6] C. L. Monma and S. Suri. Transitions in geometric minimum spanning trees. *Discrete Comput. Geom.*, 8:265–293, 1992.