Constrained Delaunay Triangulation

November 6, 2002

Abstract In this paper we generalize the concept of Constrained Delaunay Triangulation for plane graph restricted to quad and introduce a novel proof of its existence and uniqueness [1]. We also present a divide-and-conquer optimal $O(n \log n)$ algorithm in detail.

Keywords *plane graph restricted to quad, constrained delaunay triangulation, constrained hull, existence, uniqueness, divide-and-conquer, algorithm, degeneracy*

1 Background

Delaunay triangulation is a fundamental geometric construct in computational geometry. It is a triangulation of a planar point set with the property that for every triangle no point falls inside of the circum-circle of it. Delaunay triangulation as well as its dual, Voronoi Diagram, and their variations have been extensively studied[3]. Up to now, an upper time bound of $O(n \log n)$ is known using either divide-and-conquer [6] or plane-sweep line algorithm [2].

A Constrained Delaunay Triangulation is intended to resemble the ordinary Delaunay Triangulation of a planar point set as much as possible, with the additional constraint that it must contain certain prescribed edges [1, 4]. For a formal definition please refer to section 2.

D.T.Lee and a.k.lin first introduced Generalized delaunay triangulation for planar graphs, gave a proof of its existence and uniqueness and presented an incremental $O(n^2)$ algorithm for this problem. L.P. Chew then used the term Constrained Delaunay Triangulation instead and described a divide-and-conquer optimal $O(n \log n)$ algorithm roughly by introducing virtual infinite vertices.[1]. Raimund Seidel presented a sweep-line algorithm with also $O(n \log n)$ worst time cost[4]. Here we generalize the concept of Constrained Delaunay Triangulation for plane graph restricted to quad without using virtual infinite vertices and introduce a novel proof of its existence and uniqueness [1]. We also present a divide-and-conquer optimal $O(n \log n)$ algorithm in detail. The divide-and-conquer algorithm may gain more benefits from distributed computing than sweep-line algorithm.

[1]If there is no co-circular degeneracy.

The organization of the paper is as follows. In section 2, we generalize some terms in delaunay triangulation for constrained case of graph in quad. In section 3, we give the proof for the existence and uniqueness. The algorithm plus the proof of its correctness is described in section 4 . In section 5, we draw conclusion.

2 Preliminaries

2.1 Terms and denotations

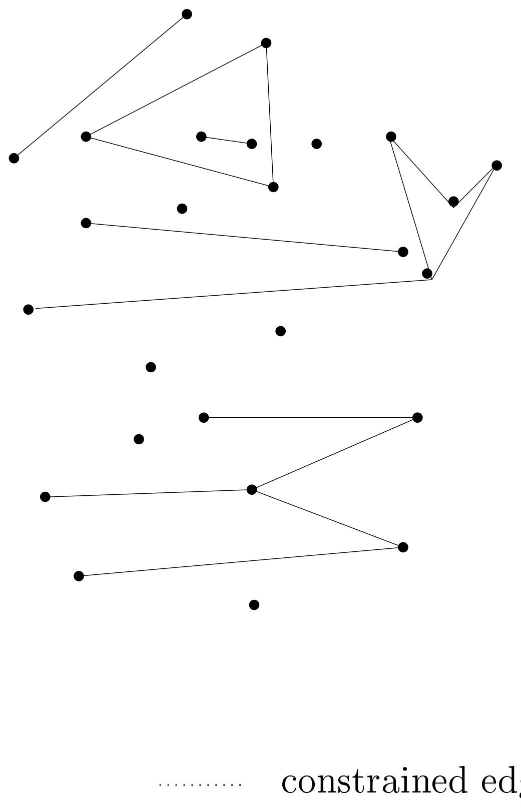

Figure 1: Example of plane Graph G

A **plane graph** $G = (V, E)$ such that $V \in \mathcal{R}^2$ is a set of points, E is a set of open [2] line-segments whose endpoints belong to V and the elements of $V \cup E$ are pairwise disjoint. See Figure 1. We call these points **vertices** and these line-segments **edges**. More than 3 vertices lie on a common circum-circle is called a **co-circular degeneracy**.

[2]All the line-segments discussed in this paper are open.

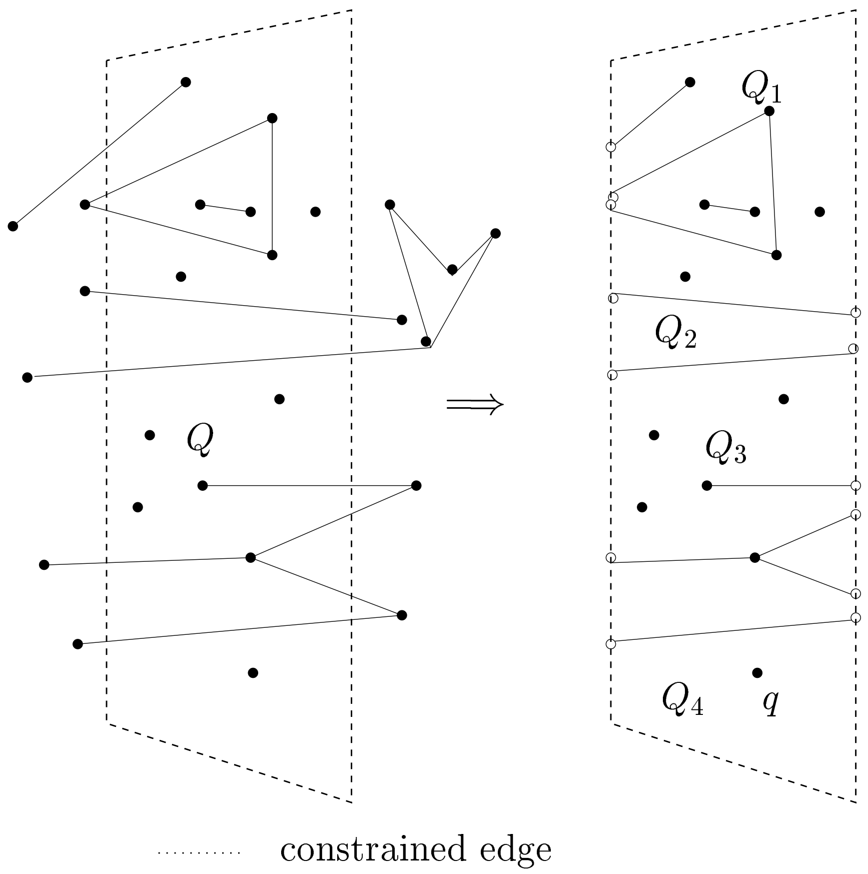

Figure 2: Example of graph G restricted to Q.

A **quad** is an open [3] convex quadrilateral region whose two sides are vertical line-segments which we call **walls**. See Figure 2. A **quad-graph** $Q(G) = (V', E')$ is G restricted to quad Q such that $V' = V \cap Q$ and $E' = \{e \cap Q : e \in E\}$. So any endpoint of edge in E' is either in V' or on the boundary of Q. The edge in E' is respectively called **line-edge, ray-edge and segment-edge** if its two, one and zero endpoint(s) are on the boundary of Q. *In this paper we only discuss those quad-graphs whose endpoints on the boundary are all on walls.* A quad is **active** if $V' = \emptyset$, **simple** if there is no line-edge. So each quad Q can be divided by its line-edges into a list of simple quads . In Figure 2 they are Q_i where $i = 1, \ldots, 4$. We call the list of active ones in above list the **active quad list** of Q. In Figure 2 it is $\{Q_1, Q_3, Q_4\}$.

[3]Open means does not contain boundary.

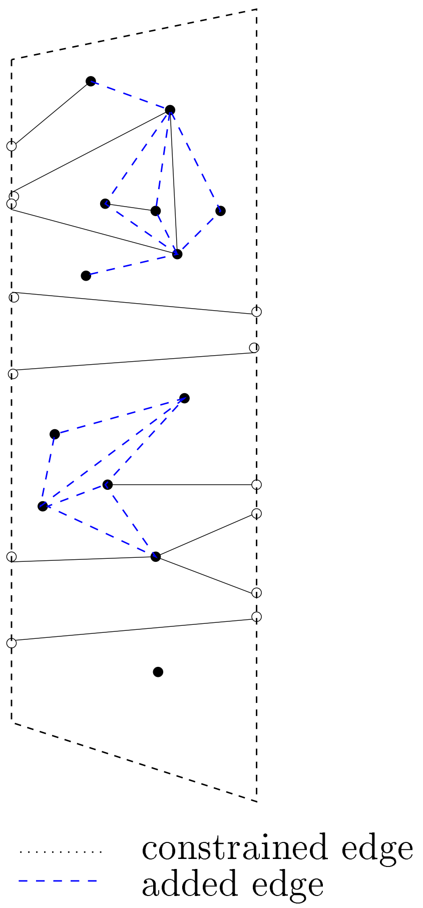

Figure 3: Triangulation for $Q(G)$.

A **(constrained) triangulation** of quad-graph $G = (V, E)$, is $T(G) = (V, E')$ such that $E \subseteq E'$ and E' is a maximum set of disjoint edges whose endpoints are in V. Each edge in E persists in the constrained triangulation. We call it a **constrained edge**. See Figure 3.

Two vertices are **visible** from each other if the line-segment between them does not intersect any constrained edge [4]. We also say these two vertices are **inter-visible** and the line-segment is a **visibility edge**. If a vertex is not visible from any other vertices, it is called an **isolated vertex**. Vertex q in Figure 2 is an isolated vertex.

Given a convex region R, we use $V(R, v)$ to represent the set of vertices in R which are visible from vertex v in R. Given a pair of vertices p, q, let

[4]The two endpoints of a constrained edge are considered visible to each other

$V(R, pq) = V(R, p) \cap V(R, q)$. A vertex r in $V(R, pq)$ which maximizes $\angle prq$ is called an **optimal visible vertex** for p, q. R is **empty** for v if $V(R, v) = \emptyset$. An **empty circle** for an edge pq is a circle O which has pq as a chord and satisfis $V(O, pq) = \emptyset$. We also say this circle is an empty circle for this pair of endpoints. A **candidate edge** is any visibility edge that has an empty circle.

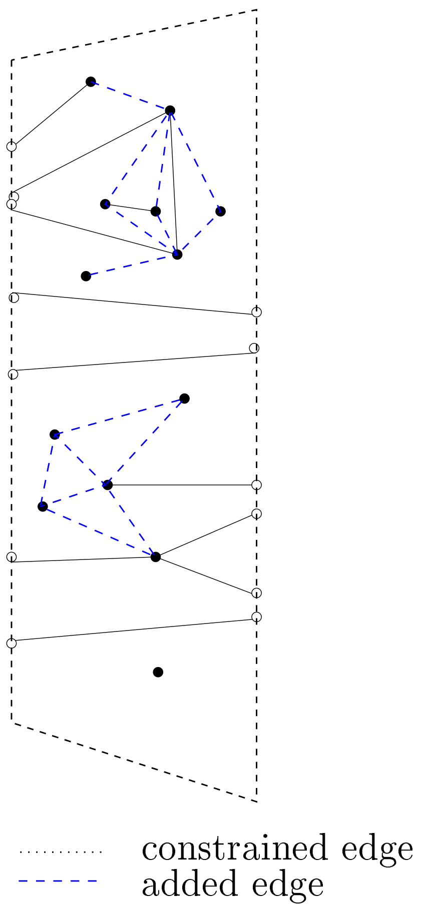

Figure 4: CDT for $Q(G)$.

A **constrained delaunay triangulation(**CDT**)** is a constrained triangulation such that each edge is either a constrained edge or a candidate edge. See Figure 4. Obviously $CDT(Q(G))$ equals to the union of $CDT(Q_i(G))$ for all Q_i which is in the active quad list of Q. Because first we only need to care about simple active quads; secondly any quads in the list other than the quad one vertex belongs to is empty for this vertex so we can do CDT locally in each Q_i.

The **constrained hull** of a quad-graph G is denoted by $H(G) = (V', E')$, where E' is the set of visibility edges which each has an empty half-plane aside of the line through it for each of its endpoints; V' is the set of vertices which contains all endpoints of edges in E' plus all isolated vertices. See Figure 5. We call each edge in E' a **hull edge** and the above corresponding half-plane an **empty half-plane** for it. Obviously every hull edge is a candidate edge since it is a visibility edge and we can always find an empty circle whose center is in its empty half-plane.

In the paper, we use v, p, q, r, s to represent vertices; V to represent a set of vertices; pq to represent edge with p, q as endpoints; E to represent a set of edges; $\ell(pq)$ to represent the directional line through pq from p to q; λ to represent a vertical line; γ to represent a ray; $O(pqr)$ to represent the circle where p, q, r lies on its circum-circle; $\angle prq$ to represent the convex angle with r as the corner; $\triangle prq$ to represent angle with p, r, q as vertices; P, Q to represent quads; and R, S, W, T to represent convex regions as well as wedges or half-planes.

3 Proof of CDT's Existence and Uniqueness

We first give several general facts as lemmas which will be used later.

LEMMA 1 *If pq is a candidate edge, r, s are visible from both p, q and lie on opposite sides of $\ell(pq)$ then s does not lie in O_{prq}.*

Proof. See Figure 6. By contradiction, suppose s lies in O_{prq}, then any circle with pq as a chord will contain at least one of r, s, this contradicts with the fact that pq is a candidate edge. **Q.E.D.**

LEMMA 2 *Given candidate or constrained edge pq, suppose $\ell(pq)$ splits the plane into two half-planes R, S, for either half-plane, say R, if $V(R, pq) \neq \emptyset$, then we can find a vertex r from $V(R, pq)$ such that pr, qr are also candidate edges or constrained edges.*

Proof. See Figure 7. We claim that the optimal visible vertex in R for pq is the r we seek.

Since $\angle prq$ is maximized, no vertex of $V(R, pq)$ lies in O_{prq}. Otherwise suppose r' lies in, then $\angle pr'q > \angle prq$, which contradicts. Note that this only proves $V(R, pq) = \emptyset$, while not necessarily $V(R, p) = V(R, q) = \emptyset$.

We claim that O_{prq} is an empty circle for $pr(qr)$ if it is not a constrained edge:

Observe that pr, qr, pq divides O_{prq} into four parts: $\triangle prq, W_{pq}, W_{pr} and W_{qr}$. Firstly, it is obvious that $\triangle prq$ is emtpy for p, q, r since r is visible from both p and q . Thus $V(\triangle prq, pr) = V(\triangle prq, qr) = \emptyset$

Secondly, we claim $W_{pq} = S \cap O_{prq}$ is emtpy for r. If pq is a constrained edge, since any vertices in W_{pq} is not visible from r, so W_{pq} is empty for r thus empty for pr, qr.

If pq is a candidate edge, using lemma 1, any vertex in $V(S, pq)$ is not in O_{prq} thus also not in W_{pq}. If there is some vertex s in W_{pq} which is not visible to both p, q, thus there exists some constrained edge which crosses W_{pq} and lies in between s and r, so W_{pq} is empty for r thus empty for pr, qr.

For either case, $V(W_{pq}, pr) = V(W_{pq}, qr) = \emptyset$.

Finally, we claim if $pr(qr)$ is not a constrained edge, O_{prq} are empty for $pr(qr)$:

For example, suppose pr is not a constrained edge, in a way similar to above, we can prove W_{qr} is empty for p thus empty for pr. $V(W_{qr}, pr) = \emptyset$.

For W_{pr}, if there is some vertex s' in W_{pr} which are visible from p while not q, obviously there are must be a constrained edge crossing W_{pr} through p and separating r and s'. So W_{pr} is empty for pr. $V(W_{qr}, pr) = \emptyset$.

So all together, $V(O_{prq}, pr) = \emptyset$ if pr is not a constrained edge.

This proved $pr(qr)$ is either a constrained edge or candidate edge. **Q.E.D.**

Corollary 3 *If pq is a candidate edge or constrained edge and vertex r is the optimal visible vertex for pq, then each of pr, qr is a candidate edge or constrained edge.*

Theorem 4 *Given quad-graph $G = (V, E)$, there exists a $CDT(G)$.*

Existence

Proof. We already know constained hull forms a set of simple polygons which satisfy:

1. the region outside of them need not to be triangulated

2. any edge on or outside of it is a candidate edge or constrained edge.

We call these kind of simple polygons **safe polygons**.

Then we claim that we can always find a CDT by repeatly finding candidate or constrained edges which connects to the inside hulls of current safe polygons, see Figure 8. This forms smaller safe polygons, so finally we get a triangulation. We only need to prove that we can always find such edges before we get a triangulation. Actually, using lemma 2, it is obviously true. **Q.E.D.**

Lemma 5 *No candidate edge will intersect if there is no co-circular degeneracies.*

Proof. See Figure 9. Otherwise, suppose candidate edges pq, rs intersects, $prqs$ forms a convex quadrilateral. Since pq is a candidate edge, we can find an empty circle for pq such that r and s does not lie in it. Since there is no 4-point co-circular,

$$\angle prq + \angle psq < \pi$$

Similarly for candidate edge rs, we have

$$\angle rps + \angle rqs < \pi$$

So:

$$\angle prq + \angle psq + \angle rps + \angle rqs < 2\pi$$

This contradicts the fact that for any convex quadrilateral $prqs$,

$$\angle prq + \angle psq + \angle rps + \angle rqs = 2\pi$$

. **Q.E.D.**

COROLLARY 6 *All candidate edges are in CDT if there is no co-circular degeneracy.*

Proof. Otherwise suppose two vertices p, q such that pq is a candidate edge while pq is not in CDT. Then pq will intersect some edge rs in some CDT. Since p, q is inter-visible, they can not intersect a constrained edge. But if rs is a candidate edge, it contradicts lemma 5. **Q.E.D.**

COROLLARY 7 *If pqr is a triangle in CDT, then r is the optimal visible vertex for pq.*

Proof. Using corollary 6 and corollary 3. It is obviously true. **Q.E.D.**

THEOREM 8 *Given quad-graph $G = (V, E)$ where there is no co-circular degeneracy, there is* **one and only one** *CDT for it.*

Uniqueness

Proof. Using corollary 6, every candidate edge and constrained edge is in CDT if there is no co-circular degeneracy. While any edge in CDT is either a candidate edge or constrained edge, the CDT is unique. **Q.E.D.**

4 Algorithms

Given a plane graph $G = (V, E)$, we construct its CDT using divide and conquer algorithm with time cost $O(n \log n)$ where n is the number of vertex in V.

4.1 Overall Design

---

1. Divide

   Construct a quad Q such that $Q(G) = G$. The problem turns into construct the CDT for quad graph $Q(G)$. Divide Q by $n - 1$ vertical lines into a list of consecutive quads which each contains a single vertex in V.

2. Conquer

   We proceed in stages. Initially at stage 0, since there is only one vertex in each quad, we have got the CDT for each quad.

   From each stage i to $i+1$, we merge disjoint pairs of adjacent quads and get the CDTs for merged quads in $O(n)$ using algorithm described later, which thus reduing the number of quads by half. We stop then there is only one quad, so finally we get the CDT for whole graph and there are at most $\log n$ stages.

4.2 Merge Quads

First we give some terms helpful for describing quads merging:

A **hour glass hull** is a simple polygon where there exist two edges whose endpoints are connected directly or by concave edge chain. These two edges are called **base edges**. See Figure 10.

Suppose a quad Q is separated by a vertical line called **separate line** into two slender quads Q_l, Q_r and QL_l, QL_r are respectively the active quad list for them. $Q(G)$'s segment-edges which intersects the separate line are called **spanning edges** and $Q(G)$'s spanning hull edges are called **bridges**. For any two quads which are respectively from QL_l and QL_r, the shared line-segment of their walls is called a **window**. It is obvious that for each window, there exists exactly one pair of bridges for $Q(G)$ [5]. See Figure 11.

The portion of window which is bounded by a pair of bridges may be divided by the spanning constrained edges into what we call **subwindows**. It is also obvious that the spanning constrained edges and bridges plus $Q_l(G), Q_r(G)$'s hull-edges forms a list of consecutive hour glass hulls which each corresponding to each subwindow. See Figure 12.

Then we give some lemmas which guarantes the correctness of the algorithm:

LEMMA 9 *Given a quad-graph $Q(G) = (V, E)$, denote E_s as the set of edges which each is either a spanning candidate edge or a spanning constrained edge. Suppose λ separates Q into Q_l, Q_r, a triangulation $T = (V, E')$ is $CDT(Q(G))$ if $E_s \subseteq E'$ and any edge in $E_c = E' - E_s$ is in either $CDT(Q_l(G))$ or $CDT(Q_r(G))$.*

Proof. See Figure 13. Obviously any spanning edge in T is in E_s. We claim an edge in E_c which forms a triangle with two edges in E_s is in $CDT(Q(G))$.

Denote the set of edge of this kind E'_s. Suppose $pq \in E'_s$ and $pr, qr \in E_s$, if pq is a constrained edge, then it is trivial; If not, suppose pq is not a candidate edge for $Q(G)$, see Figure 14, there must be some spanning candidate edge rs intersect pq, which contradicts that all spanning candidate edges are in E_s. So any edge in E'_s is in $CDT(Q(G))$.

[5]The two bridges of the pair may identical

Also we know any hull-edge is in CDT, it is easy to see that the hull-edges for $Q_l(G), Q_r(G)$ and edges in E'_s form a set of safe polygons [6] for $Q(G)$ which do not intersect λ.

For any edge on a safe polygon, say pq, assume p, q are both in Q_l, suppose $\triangle psq$ is the triangle in the polygon, since pq, ps, qs are all in $CDT(Q_l(G))$, using corollary 7, s is the optimal visible vertex for pq on one side of $\ell(pq)$ in Q_l.

We can prove s is the optimal visible vertex s' for pq on this side of $\ell(pq)$ in Q. Suppose s' is in Q_r otherwise, see Figure 15, using corollary 3, ps', qs' are both in E_s, which contradicts pqs is in T. So s is the optimal visible vertex in Q.

Since pq is in $CDT(Q(G))$, using corollary 3, ps, qs are also in $CDT(Q(G))$. So we get a smaller safe polygon.

By decreasing the size of safe polygons incrementally, we can prove any edge in E_c is in $CDT(Q(G))$. So T is a $CDT(Q(G))$. **Q.E.D.**

COROLLARY 10 *If edge pq is a candidate edge for $Q(G)$ and p, q are both in $Q_l(Q_r)$, pq is also a candidate edge for $Q_l(G)(Q_r(G))$.*

Now the algorithm:
Given the CDTs for $Q_l(G), Q_r(G)$, we construct the CDT for $Q(G)$:

---

1. Find bridges

   Add the bridges of $Q(G)$. Since we have got the CDTs for $Q_l(G), Q_r(G)$, we can get the constrained hulls for them. Using the constrained hulls merging algorithm described later, we can find the bridges in $O(m)$, where m is the total number of vertex in Q. Thus at each stage the time complexity is $O(n)$.

2. Finding spanning candidate edges

   For each subwindow, we add the spanning candidate edges for $Q(G)$ and delete any edge intersecting them. This can be also done in $O(m)$ using the hour glass hull re-triangulation algorithm described later. Thus at each stage the time complexity is $O(n)$.

Finally we get a triangulation which contains all spanning candidate or constrained edges and where any other edges are in $CDT(Q_l(G))$ or $CDT(Q_r(GJ))$. Using lemma 9, we know we get the CDT for $Q(G)$. And the overall algorithm cost $(O(n) + O(n)) * \log n = O(n \log n)$.

4.3 Merge Constrained Hulls

The constrained hulls for $Q_l(G), Q_r(G)$ are the union of small constrained hulls for active quads in Q_l, Q_r. The problem of finding constrained hull for $Q(G)$

[6] Refer to theorem theorem 4.

turns into finding the pair of bridges for each window which connects these small hulls:

For a window shared by **simple active** quads P, Q respectively on the left and right, given $H(P(G))$ and $H(Q(G))$, we want to find the pair of bridges in $O(m_1 + m_2)$ where m_1, m_2 are respectively number of vertex in P, Q.

Let $p_0, p_1, \ldots, p_l, p_0$ be the list of vertices we get by a counter-clockwise circular traversal of $H(P(G))$. Notice that p_i may identical to p_j for different i, j and $l < 2m_1$. Similarly let $q_0, q_1, \ldots q_r, q_0$ be the list of vertices we get by a clockwise circular traversal of $H(Q(G))$.

Without loss of generality, suppose we want to find the lower bridge,say pq. Then p, q are inter-visible and there is no vertex below $\ell(pq)$ visible to both p, q.

For any pair (i, j) where $\ell(p_i q_j)$ has positive slope, define $W(i, j)$ as the convex wedge bounded by the horizontal ray $\gamma(p_i)$ emanating to the right of p_i and by the ray from p_i away from q_j. See Figure 16. Similarly, define $S(i, j)$ as the convex wedge bounded by $\gamma(p_i)$ and the ray from p_i toward q_j; define $T(i, j)$ as the convex wedge bounded by the vertical ray emanating to the lower of q_j and by the ray from p_i toward q_j.

A pair (i, j) to said to be a **candidate pair** if:

1. The line-segment connecting p_i, q_j does not intersect any edge in Q.

2. No vertex of P is visible from q_j in $W(i, j)$.

3. No vertex of Q is visible from p_i in $T(i, j)$.

Notice if edge $p_i p_{i+1}$ does not lie in $S(i, j)$, no vertex which is visible from both p_i, q_j lies to the lower side of $\ell(p_i q_j)$, then $p_i p_j$ is the lower bridge we seek.

First we can find some pair as $(0, 0)$ from which can determine the sign of $\ell(p_l q_r)$'s slope.

For cases where P, Q's bottom sides are segments of the same constrained edge, see Figure 17, we mark the lowest vertex in P, Q as p_0, q_0. It is obvious that sign of $\ell(p_l q_r)$' slope is same to $p_0(q_0)$'s.

For other cases, without loss of generality, suppose the bottom side of P is in fact a portion of a constrained edge which has one endpoint, say s, in Q. See Figure 18. We still mark the lowest vertex in P as p_0 while mark the lowest vertex in Q on the left side [7] of s as q_0. Also, the sign of $\ell(p_l q_r)$'s slope should be same to $p_0 q_0$'s.

Second, without loss of generality, suppose the slope of $\ell(p_0 q_0)$ is positive, given the pair $(0, 0)$, we find the initial candidate pair $(0, j)$. This can be done by repeatly increment $j \leftarrow j+1$ from $j = 0$ until q_{j+1} lies to the left of $\ell(p_i q_j)$ or some constrained edge $q_j r$ intersect $\ell(p_i q_{j+1})$. This process to find a candidate pair restricting to some p_i is called **pair matching**.

Given the initial candidate pair, we repeatly increment $i \leftarrow i + 1$ from $i = 0$ and perform pair matching to find pair (i, j). As you can see in Figure 19, the new pair (i, j) is also a candidate pair since it satisfies those three properties. We stop when p_{i+1} is not in $S(i, j)$. Termination is clear as each new candidate

[7] Including vertex s.

pair (i, j) involves a larger i-index than its predecessor. Moreover, the time is linear because we never backup on i or j. In a similar way, we can find the higher bridges in $O(l+r)$, Thus we can find the bridges in time cost $2*O(l+r)$, still $O(m_1 + m_2)$.

We now present the algorithm. This uses the primitive RightTurn(p, q, r) that returns true iff r lies to the right of the directed line $\ell(pq)$ and sub-routine NoIntersection(q, p, r) that returns true iff pr does not intersect any constrained edge connecting q.

```
Input: $P = (p_0, \ldots, p_l)$ and $Q = (q_0, \ldots, q_r)$.
Output: $(i, j)$ where $(p_i, q_j)$ is a lower bridge.
    1. $j \leftarrow 0$;
        $\{$Find initial candidate pair $\}$
    2. while (NoIntersection$(q_j, p_0, q_{j+1})$ AND RightTurn$(p_0, q_j, q_{j+1})$)
        $j \leftarrow j + 1$;
        $\{$Now $(0, j)$ is a candidate pair$\}$
    3. $i \leftarrow 0$;
        $\{$Main Loop$\}$
    4. while RightTurn$(p_i, q_j, p_{i+1})$
    4.1  $i \leftarrow i + 1$;
    4.2  while (NoIntersection$(q_j, p_i, q_{j+1})$
         AND RightTurn$(p_i, q_j, q_{j+1})$)
             $j \leftarrow j + 1$;
         $\{$When the inner while-loop terminates, $(i, j)$ is candidate pair$\}$
    $\{$When the outer while-loop terminates, $(i, j)$ is a bridge$\}$
    5. return$((i, j))$
```

4.3.1 Hour Glass Hull Re-triangulation

Given a hour glass hull whose base edges are spanning candidate or canstrained edges, we want to find other spanning candidate edges in between and delete edges intersecting them. This problem can be turned into repeatly adding next spanning candidate edge above current lower base edge and replacing it as a new base edge until the new base edge is the upper base edge.

Suppose we want to find the new base edge above base edge pq where $p \in P$ and $q \in Q$. Without loss of generality, we can assume the optimal visible vertex r for pq is in P. Using corollary 3, qr is the next base edge. Using corollary 10, pr is in $CDT(P(G))$. So we only need to find pr in those edges already in $CDT(P(G))$.

Let $p_1, p_2, \ldots$ be the vertices, listed in counter-clockwise order, that are connected to p in $CDT(P(G))$. Here, p_1 is chosen to be the first vertex among those connected to p that lies above the base edge. Similarly, let $q_1, q_2, \ldots$ be the vertices, listed in clockwise order, that are connected to q in $CDT(Q(G))$. Again, q_1 is the first vertex among those connected to q that lies above the base edge. Then we only need find r in $\{p_1, p_2, \ldots\} \cup \{q_1, q_2, \ldots\}$.

Lemma 11 *Let $pqrs$ be a convex quadrilateral, and pr is a candidate edge for $\{p, q, r, s\}$. Suppose O is a circle with p, q as a chord. If r lies outside O then s must lie outside O.*

Proof. Refer to Figure 20. Let P, Q, R, S be the angles at vertices p, q, r, s in the quadrilateral $pqrs$. By definition of candidate edge, $Q + S \leq \pi$. While $P + R + Q + S = 2\pi$, so

$$P + R \geq Q + S.$$

Let pr intersect O at r', and let R', Q', S' be the angles at r', q, s in the quadrilateral $pqr's$. Then, since $R' > R, Q' < Q, S' < S$, we have

$$P + R' > Q' + S'.$$

By way of contradiction, suppose s lies inside O. Then let s' be the point on O such that ps' contains s. Let S'', R'' be the angles at s', r' in the quadrilateral $pqr's'$. Since $S'' < S'$, $R'' > R'$, we conclude

$$P + R'' > Q' + S''.$$

This is a contradiction, because $pqr's'$ lies on O and hence the sum of their opposite angles must be equal. **Q.E.D.**

Corollary 12 *If q_{i+1} lies outside $O(pqq_j)$, then for all $j \geq j + 2$, q_j must lie outside. We say qq_j is safe.*

Using this corollary, our basic algorithm to find next base edge is as follows, see Figure 21:

1. Repeatedly increment $i \leftarrow i + 1$ until pp_i is a constrained edge or is safe.
2. Repeatedly increment $j \leftarrow j + 1$ until qq_j is a constrained edge or is safe.
3. Choose the candidate edge for quadrilateral pqq_jp_i as the next base edge and delete all edges intersecting it.

In fact, we can improve this algorithm since we have following lemma:

Lemma 13 *If some edge is not safe, say pp_i, it is not a candidate edge for $Q(G)$.*

Proof. If pp_i is a constrained edge, it is trivial. If not, then p_{i+1} is in $O(pqp_i)$ suppose s is the optimal visible vertex for pp_i on the opposite side of p_{i+1}, then $O(pp_is)$ must contain p_{i+1}, using lemma 1, pp_i is not a candidate edge. **Q.E.D.**

So whenever we find an edge which is not safe, we can delete it.
The improved algorithm is as follows:

1. Repeatedly delete pp_i and increment $i \leftarrow i + 1$ until pp_i is a constrained edge or is safe.

2. Repeatedly delete qq_j and increment $j \leftarrow j+1$ until qq_j is a constrained edge or is safe.

3. Choose the candidate edge for quadrilateral pqq_jp_i as the next base edge.

We know a planar graph with $m \geq 3$ vertices has at most $3m-6$ edges[3]. So we will perform at most $3m-6$ safe edge testing, edge deleting and edge adding. All together the time cost is $O(m)$.

5 Conclusion

We have ...may improve on...

References

[1] L.P. Chew. Constrained delaunay triangulations. In *Proc. 3rd Ann. ACM Sympos. Comput. Geom.*, pages 215–222, 1987.

[2] S.J. Fortune. A sweepline algorithm for vonoroi diagrams. *Algorithmica*, 1987.

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

[4] R. Seidel. Constrained delaunay triangulations and voronoi diagrams with obstacles. Technical Report 260, IIG-TU Graz, Austria, 1988.

[5] C.A. Wang and L. Schubert. An optimal algorithm for constructing the delaunay triangulation of a set of line segments. In *Proc. 3rd Ann. ACM Symbos. Comput. Geom.*, pages 223–232, 1987.

[6] Chee K. Yap. An $o(n \log n)$ algorithm for the voronoi diagram of a set of simple curve segments. In *Discrete and Computational Geometry*, pages 365–393, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, USA, 1987. Springer-Verlag New York Inc.

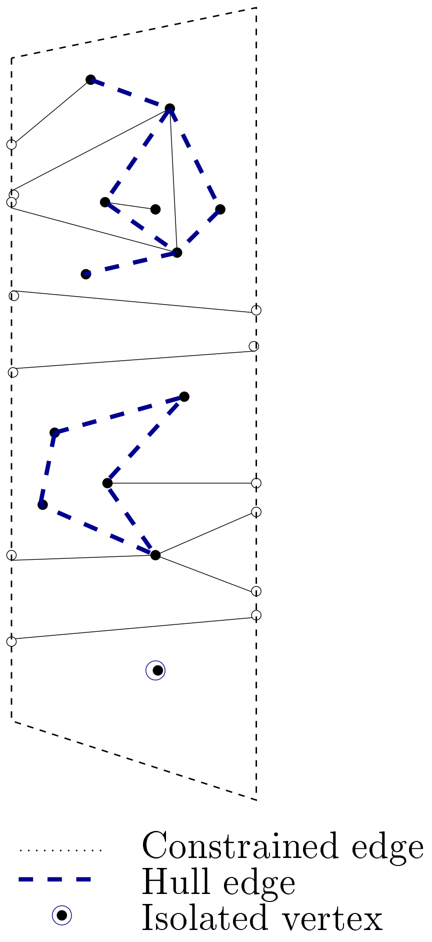

Figure 5: Constrained hulls for $Q(G)$.

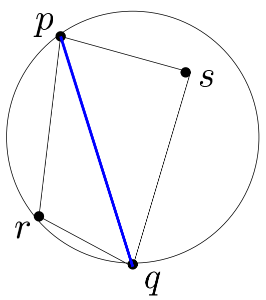

Figure 6: s does not lie in O_{prq}

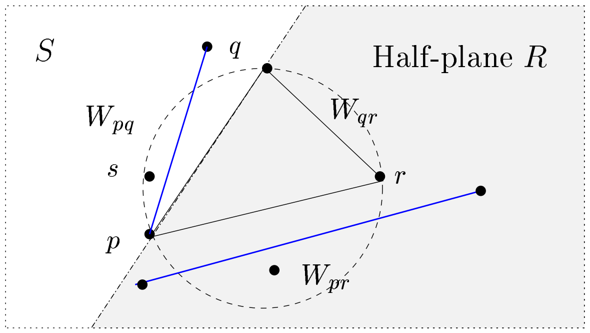

Figure 7: pr, qr each is either a candidate or a constrained edge

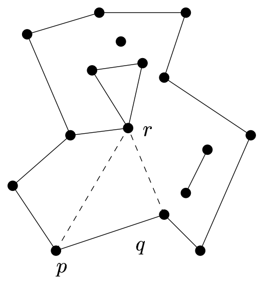

Figure 8: Shrink the size of safe polygon by replace pq with pr, qr.

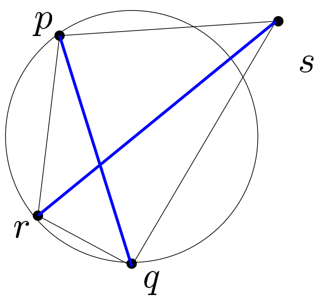

Figure 9: pq and rs can not be both candidate edges.

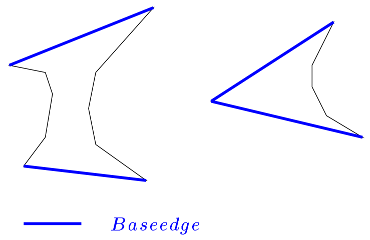

Figure 10: Two examples of hour glass hull.

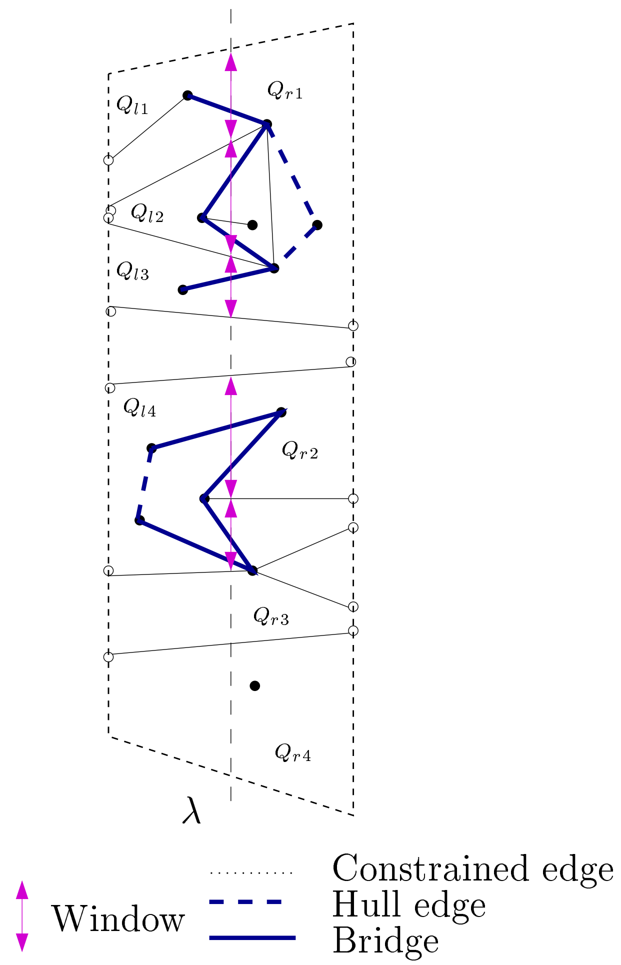

Figure 11: Q is separated by λ.

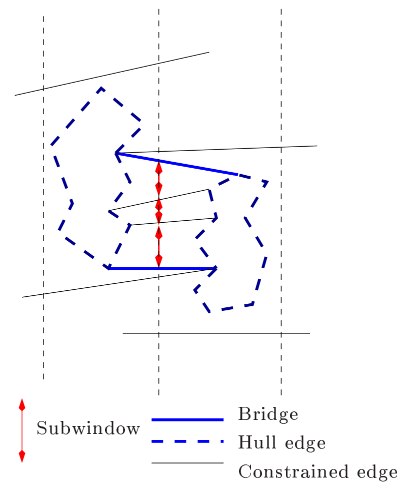

Figure 12: Subwindows between two bridges.

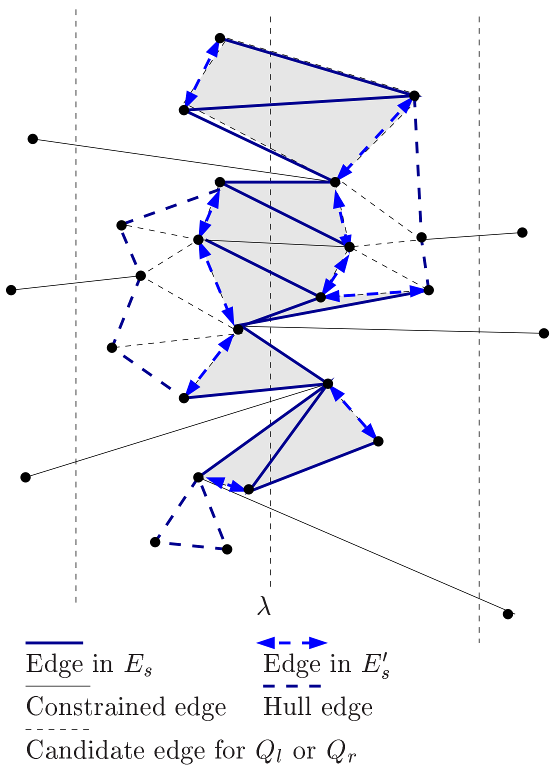

Figure 13: This triangulation is a CDT

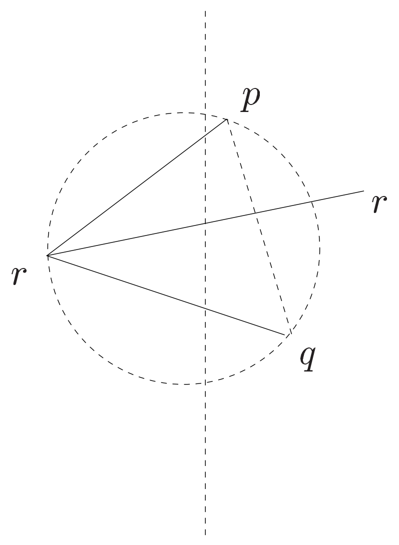

Figure 14: pq is a candidate or constrained edge

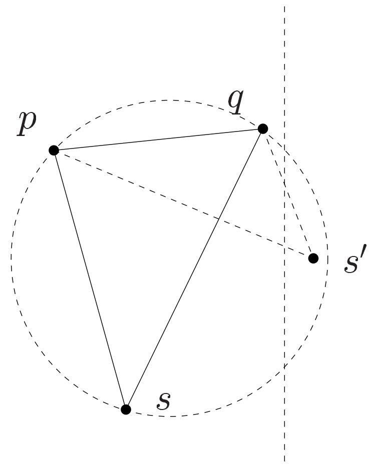

Figure 15: s is the optimal visible vertex for pq

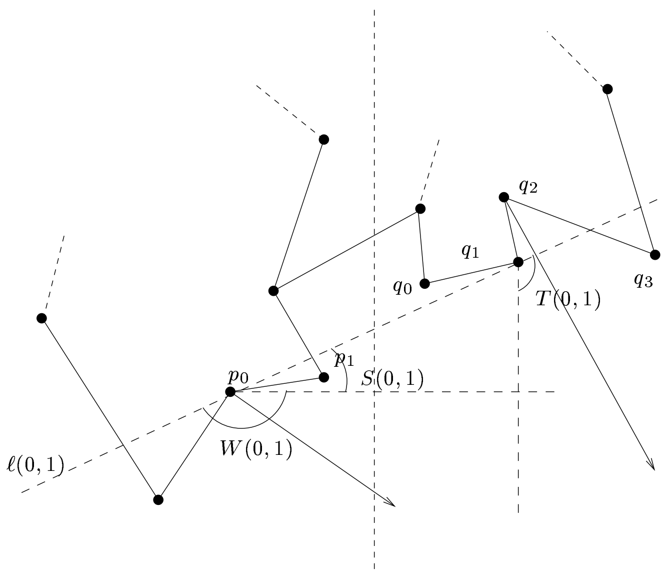

Figure 16: Wedge $W(i,j), S(i,j), T(i,j)$.

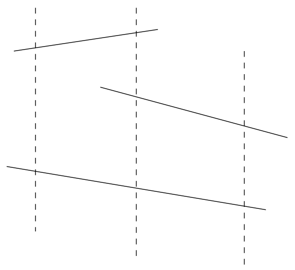

Figure 17:

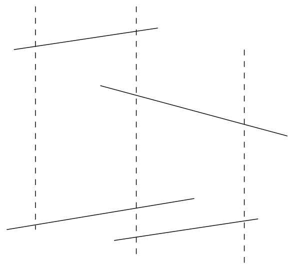

Figure 18:

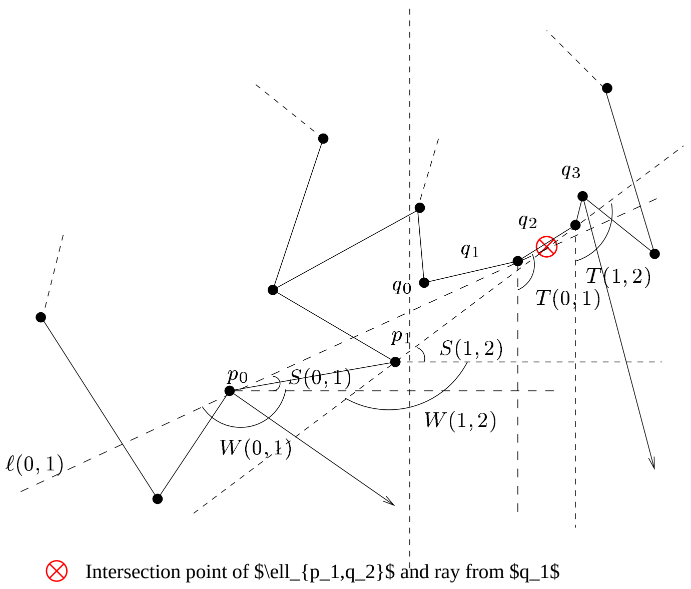

Figure 19: Advance Procedure

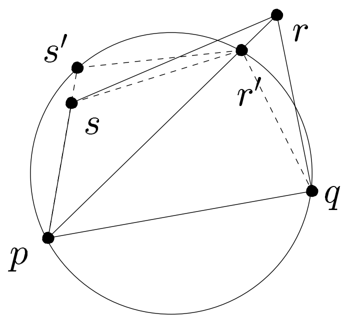

Figure 20: Convex quadrilateral $pqrs$

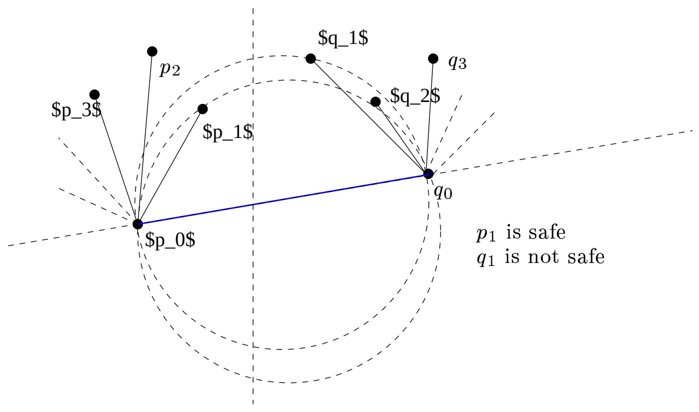

Figure 21: Finding next base edge.

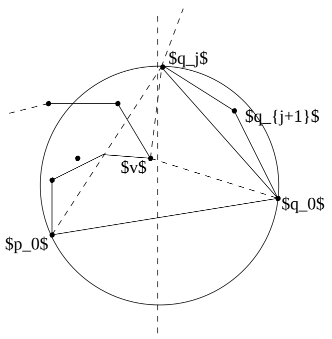

Figure 22: qq_j is not safe.