A Slow Algorithm for Computing the Gabriel Graph with Double Precision

David L. Millman Vishal Verma
Department of Computer Science, University of North Carolina at Chapel Hill*

Abstract

When designing algorithms, time and space usage are commonly considered. In 1999, Liotta, Preparata and Tamassia proposed that we could also analyze the precision of an algorithm. We present our first steps towards the goal of efficiently computing the Gabriel graph of a finite set of sites, while restricting ourselves to only double precision.

1 Introduction

Computers use finite precision arithmetic to test geometric relationships between objects. Sometimes, a computer cannot provide a sufficient number of arithmetic bits to guarantee that the tests are correct. It is natural then to analyze the number of bits required to correctly run an algorithm. One such model of analysis was proposed by Liotta, Preparata and Tamassia [**?**]. They define the degree (or arithmetic complexity) of an algorithm in terms of the arithmetic degree of its predicates. One can then attempt to minimize the degree of an algorithm, just like time and memory. This type of analysis is known as *degree-driven algorithm design*, and it tells us the amount of arithmetic precision required to run the algorithm safely. In Section **??**, we define degree and arithmetic precision more formally. Arithmetic precision is of special interest when running geometric algorithms, that assume general position, on practical datasets that are frequently close to being degenerate. To avoid these issues of instability we have been working on degree driven algorithms for constructing geometric structures. In this paper we describe our first step towards constructing the Gabriel graph robustly.

Given a finite set of sites S, an edge (s_i, s_j) with $s_i, s_j \in S$ is in the *Gabriel graph of* S if the edge maintains the Gabriel property, that is, the closed disk with diameter $\overline{s_i s_j}$ contains no points of S besides s_i and s_j. It is know that the Gabriel graph [**?**] is a subgraph of the Delaunay triangulation. Matula and Sokal [**?**] showed how to compute the Gabriel graph directly from the Delaunay triangulation in time proportionate to the number of sites in S.

Computing the Delaunay triangulation requires four times the precision of the input coordinates, and Matula and Sokal's Gabriel graph algorithm uses six-fold precision. Liotta [**?**] showed how to implement Matula and Sokal's algorithm using only two-fold precision, however, it still requires four-fold precision for computing the Delaunay triangulation. A natural question that follows is, can we compute the Gabriel graph with only two-fold precision?

The answer is yes! In Section **??** we show that we can compute the Gabriel graph with two-fold precision (albeit rather slowly).

2 Definitions and Notation

We begin by recalling how one can analyze arithmetic complexity. Assume that the coordinates of our input can be scaled to b-bit integers. Thus, we can think of the sites of S as lying on the $U \times U$ grid, notated as $\mathbb{U}$. The primitives of a geometric algorithm are called *predicates*, which are tests of the signs of multivariate polynomials with variables from the input coordinates. We say that the *degree* of a predicate is the degree of the polynomials to which it corresponds (for a degree d predicate we sometimes say it uses d-fold precision). Furthermore, we define the *degree of an algorithm* by the highest degree predicate it evaluates.

Consider, for example, testing if point q is closer to point p_1 or p_2 with $p_1, p_2, q \in \mathbb{U}$. We can write this predicate as $\text{sign}(\|q - p_1\|^2 - \|p - p_2\|^2)$. Which expands to a degree 2 polynomial, thus, this predicate is degree 2, (i.e., it uses two-fold precision). Another example (used in Section **??**) tests if the straight line path from p_1 to p_2 to q, with $p_1, p_2, q \in \mathbb{U}$, forms a counterclockwise orientation. This $\texttt{Orientation}(p_1, p_2, q)$ predicate, tests the sign of the determinant of the homogeneous coordinates of p_1, p_2 and q, and is also degree 2.

Next, recall the point/line duality [**?**] that maps a point $p = (p_x, p_y)$ to a line $p^* := (y = p_x x - p_y)$ and a line $l := y = mx + b$ to a point $l^* := (m, -b)$. The set S^* is the set of lines, dual to the set of sites of S. We notate the arrangement of the lines in S^* is $A(S^*)$ and the Gabriel graph of S as $G(S)$.

3 Arrangements of Dual Lines

It is know that for a set of line segments, defined by their endpoints, computing an arrangement requires

*[dave,verma]@cs.unc.edu

four times the input precision and computing its trapezoidation requires five times the input precision [**?**]. In this section, we show that for a set of lines, defined as duals of points, computing an arrangement and its trapezoidation can be solved with double precision.

We begin by observing that for non-parallel lines p^* and q^*, the the x-coordinate of the point $\ell^* = p^* \cap q^*$ is the slope of line $\ell = \overleftrightarrow{pq}$, which is $(p_y - q_y)/(p_x - q_x)$.

Observation 1 *The x-coordinate of the intersection of two dual lines is represented by a rational polynomial of degree 1 over degree 1.*

For three dual lines, p^*, q^*, and r^*, where p^* and q^* intersect r^*, by Observation **??** and clearing fractions, we compare the x-ordering of the intersection points with degree 2. We call this the `OrderOnALine`(p^*, q^*, r^*) predicate.

By using the `OrderOnALine` predicate in an incremental construction of an arrangement (such as [**?**, Chapter 8.3]) we achieve a degree 2 construction. Furthermore, we can use the `OrderOnALine` predicate to add the verticals into the arrangement and get its trapezoidation.

Lemma 1 *For n dual lines S^*, we can compute the arrangement of S^* and its trapezoidation in $O(n^2)$ time and degree 2.*

4 Gabriel Graphs

Next, we describe how to construct the Gabriel graph in $O(n^2)$ using degree 2, and begin by defining the primitives of our construction. Let $D(p, q)$ be the closed disk with $\overline{pq}$ as the diameter. We say that a site s *kills* the edge (p, q) if s lies in $D(p, q)$. Let m be the midpoint of $\overline{pq}$. The degree 2 predicate `IsKiller`(p, q, s) compares the squared distance between m and s and m and p to determine if s kills edge (p, q).

Given the arrangement $A(S^*)$ and a site $s_i \in S$ we would like to compute the circular orderings of the sites in $S \setminus \{s_i\}$ around s_i. Consider the line s_i^*, each vertex $v_j \in A(S^*)$ that lies on s_i^* corresponds to a line though s_i and some other site $s_j \in S$. As mentioned in Section **??**, the slope of $\overleftrightarrow{s_i s_j}$ is the x-coordinate of v_j, thus, by walking along s_i^* in $A(S^*)$ we find a set of lines, though s_i ordered by slope, which gives the circular ordering of the sites of $S \setminus \{s_i\}$ around s_i.

Constructing a circular ordering for a site is a purely topological operation on the arrangement $A(S^*)$ and uses degree 0. For each site s, computing the circular ordering takes time proportional to the number of vertices that lie on s^*, which is $O(n)$.

Lemma 2 *Given $A(S^*)$, for site $s \in S$, we can compute the circular ordering of the sites in $S \setminus \{s\}$ around s in $O(n)$ time and degree 0.*

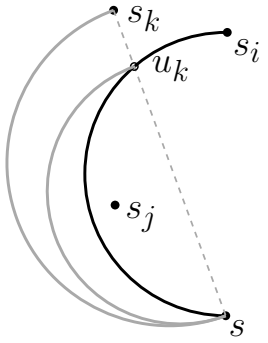

Figure 1: s_j lies in the part of $D_l(s, s_i)$ that is to the left of $s\vec{s}_k$. This part of $D_l(s, s_i)$ is a subset of $D_l(s, u_k)$, which itself is a subset of $D_l(s, s_k)$.

Once we have computed a circular ordering of $S \setminus \{s\}$ around each $s \in S$, in $O(n)$ time we can compute the Gabriel edges incident at s. The key idea behind this step is captured in Lemma **??**.

We number the circularly ordered sites in $S \setminus \{s\}$ in a counterclockwise manner starting with any $s_0 \in S \setminus \{s\}$ *i.e.* $s\vec{s}_{i+1}$ is the first ray counterclockwise from $s\vec{s}_i$ at s. Let $D_l(s, s_i)$ denote the closed semicircular disk that has $\overline{ss_i}$ as the diameter and lies to the left of $s\vec{s}_i$. We say s_j kills the edge (s, s_i) from the left if and only if s_j lies in $D_l(s, s_i)$. Then,

Lemma 3 *If s_j lies in $D_l(s, s_i)$ and $\forall k \in \{i, i+1, \ldots, j-1\}$, $s_k \notin D_l(s, s_i)$, then s_j also lies in $D_l(s, s_k)$, $\forall k \in \{i, i+1, \ldots, j-1\}$*

Intuitively, the above lemma says that if s_j is the first site (in a counterclockwise sense) that kills (s, s_i) from the left, then s_j kills all $(s, s_k), i \leq k \leq j-1$, from the left. Figure **??** gives a brief idea of the lemma and the proof.

Proof. Since $i \leq k \leq j-1$ and $s_k \notin D_l(s, s_i)$, the segment $\overline{ss_k}$ intersects the circular part of the boundary of $D_l(s, s_i)$ at some point u_k. For every point $p \in D_l(s, u_k), \angle sps_k \geq \angle spu_k > \pi/2$. Thus p also lies in $D_l(s, s_k)$. Hence $D_l(s, u_k) \subset D_l(s, s_k)$.

Let H be the closed half plane that lies on left of the line $s\vec{u}_k$. For every point $p \in D_l(s, s_i) \cap H, \angle spu_k \geq \angle sps_i > \pi/2$. Thus $(D_l(s, s_i) \cap H) \subset D_l(s, u_k)$. Using this with the subset relation from the previous paragraph we have $(D_l(s, s_i) \cap H) \subset D_l(s, s_k)$. Since s_j lies in $(D_l(s, s_i) \cap H)$ it also lies in $D_l(s, s_k)$. □

Let L be the circular linked list of sites of $S \setminus \{s\}$ circularly ordered around s. Algorithm **??** efficiently identifies sites $s' \in L$ such that the edge (s, s') is killed from the left by some site in L. Such vertices are marked *dead* by the algorithm. A similar algorithm is used to identify the sites s'' such that the edge (s, s'') is killed

from the right. The edges that are killed neither from left nor right belong to the Gabriel graph.

We now give a brief overview of Algorithm **??**. Given a site $u \in L$, we define $left\_victims(u)$ as a subset of $S \setminus \{s\}$ such that for each site $v \in left\_victims(u)$, u is the first (when walking left along the list L) site to kill the edge (s, v) from left. Lemma **??** says that the set $left\_victims(u)$ is contained in a continuous sublist of L that starts on the right of u and only contains sites w such that (s, w) is killed from left by u. This observation is used in the inner while loop of Algorithm **??** to find a sublist L_u such that: (a) each site in L_u has a killer in $(u \cup L_u)$; and (b) the union of the $left\_victims$ of the sites in $(u \cup L_u)$ is a subset of L_u. Due to (a), we know that the sites in L_u can be killed from left and hence they are marked *dead*. Due to (b), for any remaining site $v \in L \setminus L_u$, if the edge (s, v) is killed from left then v belongs to the set of $left\_victims$ of some site in $L \setminus L_u$. Thus, to find the remaining $left\_victims$, we process the smaller list $L \setminus L_u$.

Algorithm 1: KillFromLeft(S, s, L)

```
Make a copy L_left of L;
Initialize the unseen values of each site in L to true;
Initialize the dead values of each site in L to false;
u = any site in L_left;
while u→unseen do
    u →unseen = false;
    killer = u;
    current = u → right;
    while killer≠current do
        if killer ∈ D_l(s, current) then
            current →dead = true ;
            current = current → right;
        else
            killer = killer → right;
        end
    end
    L_u = the sublist of L_left, that is to the right of
    u and left of current;
    Delete L_u from L_left;
    u = u →left;
end
```

Testing if $killer \in D_l(s, current)$ uses degree 2 predicates `isKiller` and `Orientation`, thus, the above algorithm runs in $O(|L|)$ time and is degree 2.

Lemma 4 *Given the circular ordering of $S \setminus \{s\}$ around s, in $O(n)$ time and degree 2, we can find the Gabriel edges incident at s.*

For completeness, we describe the three steps for constructing the Gabriel graph of a set of n sites S with degree 2. First, compute $A(S^*)$, which by Lemma **??** takes $O(n^2)$ time and degree 2. Second, for each site $s_i \in S$ compute the circular ordering of the sites of $S \setminus \{s_i\}$, which in total, by Lemma **??**, takes $O(n^2)$ and degree 0. Third, for each site $s_i \in S$, use the circular orderings to compute the set of Gabriel edges in which s_i is a member, which in total, by Lemma **??**, takes $O(n^2)$ and degree 2.

Corollary 5 *We can compute the Gabriel graph in $O(n^2)$ time using degree 2.*

5 Conclusion and Open Problems

Even though an $O(n^2)$ construction is too slow for practical applications, Corollary **??** tells us that we can at least compute the Gabriel graph with degree 2 and do better than brute force. In contrast, we simply cannot compute the Delaunay triangulation with degree 2. Two questions follow.

Firstly, can we compute the Gabriel graph in subquadratic time with degree 2? It may be of interest to note that the grid size does not appear in the running time of the algorithm. Thus, the algorithm still terminates if we let the step size of the grid shrink to zero.

Secondly, since we cannot compute the Delaunay triangulation with degree 2, can we compute a triangulation that is in some sense close to Delaunay? With degree 2 we can compute the convex hull and Gabriel graph, but which edges should we add to complete the triangulation?

References

[1] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag New York, Inc., 3rd edition, 2008.

[2] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):pp. 259–278, 1969.

[3] G. Liotta. Low degree algorithms for computing and checking gabriel graphs. Technical report, Providence, RI, USA, 1996.

[4] G. Liotta, F. P. Preparata, and R. Tamassia. Robust proximity queries: An illustration of degree-driven algorithm design. *SIAM J. Comput.*, 28(3):864–889, 1999.

[5] A. Mantler and J. Snoeyink. Intersecting red and blue line segments in optimal time and precision. In *Discrete and Computational Geometry*, number 2098 in LNCS, pages 244–251. Springer Verlag, 2001.

[6] D. W. Matula and R. R. Sokal. Properties of Gabriel Graphs Relevant to Geographic Variation Research and the Clustering of Points in the Plane. *Geographical Analysis*, 12(3):205–222, 1980.