Simple randomized $O(n \log^* n)$ algorithms*

Olivier Devillers†

Abstract : We use here the results on the influence graph [2] to adapt them for particular cases where additional information is available. In some cases, it is possible to improve the expected randomized complexity of algorithms from $O(n \log n)$ to $O(n \log^* n)$.

This technique applies in the following applications : triangulation of a simple polygon, medial axis of a simple polygon, Delaunay triangulation of vertices of a convex polygon, Delaunay triangulation of points knowing the MST (minimum spanning tree).

1 Introduction

Some worst case optimal algorithms in computational geometry are rather complicated and very difficult to implement. An attractive alternative is to use simpler algorithms whose complexities are not worst case optimal but only randomized, i.e. when averaging over all the possible executions of the algorithm. In particular, randomized incremental algorithms suppose only that all the $n!$ possible orders to introduce the n data are evenly probable. It is important to notice that no hypothesis is done about the data themselves.

The two major techniques for incremental randomized constructions are the *conflict* and *influence graphs*. The conflict graph [4] is a bipartite graph linking the already constructed results to the data not yet inserted. The algorithms using such a structure are obviously static.

The influence graph [2] links together all the intermediate results and can be used in a semi-dynamic way.

The conflict and influence graphs solve various problems with optimal expected bounds. For example, the vertical visibility map of a set of n non intersecting line segments is computed in $O(n \log n)$ expected time. In this special case, Seidel [5], merging the two kinds of graphs, succeeds in using additional information to speed up the algorithm. More precisely, the visibility map of a simple n-gon is constructed in $O(n \log^* n)$ expected time.

This article proposes new and simpler proofs for the complexity of the conflict and influence graphs that allow to extend Seidel's technique to other applications : triangulation of a simple polygon, Delaunay triangulation of vertices of a convex polygon, medial axis of a simple polygon, Delaunay triangulation of points knowing the euclidean minimum spanning tree. For all these problems the expected complexity is $O(n \log^* n)$, and the algorithms are simple and easy to code. The two first problems admit linear deterministic solutions [3,1] but they are fairly complicated and do not yield practical algorithms ; for the others, the existence of a $o(n \log n)$ deterministic solution is still open.

†INRIA, 2004 Route des Lucioles, B.P.109, 06561 Valbonne cedex (France), E-mail : odevil@alcor.inria.fr

**This work has been supported in part by the ESPRIT Basic Research Action Nr. 3075 (ALCOM).*

2 Conflict and influence graphs

The problem must be formulated in terms of objects and regions. The *objects* are the input data of the problem, they belong to the universe of objects $\mathcal{O}$. For example $\mathcal{O}$ may be the set of the points, the lines or the hyperplanes of some euclidean space. The *regions* are defined by subsets of $\mathcal{O}$ of less than b objects. The notion of *conflict* is now introduced : an object and a region may be, or not, in conflict. If F is a region, the subset of $\mathcal{O}$ consisting of the objects in conflict with F is called the *influence range* of F.

Now, the aim is to compute for a finite subset $\mathcal{S}$ of $\mathcal{O}$, the regions defined by the objects of $\mathcal{S}$ and without conflict with objects of $\mathcal{S}$; such a region is called an *empty* region of $\mathcal{S}$. The requested result is supposed to be exactly the set of empty regions or easily deductible from it.

Many geometric problems can be formulated in that way. The vertical visibility map of line segments is a set of *empty* trapezoids. The Delaunay triangulation of points is a set of triangles with *empty* circumscribing balls. A visibility graph of a set of line segments is a set of *empty* triangles.

The conflict graph

Clarkson and Shor [4] developped some algorithms based on a structure called the *conflict graph*. This graph is a bipartite graph between the empty regions of a subset $\mathcal{S}'$ of $\mathcal{S}$, and the other objects in $\mathcal{S} \setminus \mathcal{S}'$. A region and an object are linked together if they are in conflict. Thus all the conflict relationships are stored in the conflict graph and can be used in the algorithm.

The process is initialized with $\mathcal{S}' = \emptyset$. There is a unique empty region ε defined by 0 objects and each object of $\mathcal{S}$ is in conflict with ε. At each step, an object O of $\mathcal{S} \setminus \mathcal{S}'$ is added to $\mathcal{S}'$. All the regions in conflict with O are known, these regions do not remain empty after the insertion of O and must be deleted. The new empty regions defined by O (and other objects of $\mathcal{S}'$) are created and the conflicts involving these new regions are

computed to replace the conflicts involving the deleted regions. When $\mathcal{S} = \mathcal{S}'$, the conflict graph is exactly the set of empty regions of $\mathcal{S}$ which is exactly the result.

For the randomized analysis, the points of $\mathcal{S}$ are supposed to be added to $\mathcal{S}'$ in random order.

The influence graph

The conflict graph gives immediately the regions in conflict with the new object, but its design itself requires to know all the objects at the beginning of the execution. The algorithms using such a structure are intrinsically static.

The influence graph [2] is a location structure for the determination of conflicts. The nodes of this graph are the regions having been empty at one step of the incremental construction. This graph is rooted, directed and acyclic ; the leaves of this graph are the currently empty regions. The influence graph satisfies the following property : the influence range of a region is included in the union of the influence ranges of its parents.

The influence graph is initialized with a single node : the root, associated to the region ε whose influence range is the whole universe of objects $\mathcal{O}$. When a new object O is inserted, the above property allows to traverse all the regions of the graph in conflict with O ; all the empty regions in conflict are reported. These regions do not remain empty (they contain O) but they still are nodes of the influence graph. Then, as for the conflict graph, the new empty regions are computed, and are linked to the already existing regions in order to ensure the further determination of conflicts ; they are linked such that the property is verified.

For the sake of simplicity, we will suppose that the number of sons of a node is bounded. This hypothesis is not really necessary and can be relaxed ; but it is fulfilled by a large class of geometric problems and allows to express the results in a simple way.

3 Analysis

The proofs of lemmas are omitted in this abstract. Let us just recall here, that all results are randomized, that is the $n!$ possible orders for the insertion of the n objects in $\mathcal{S}$ are evenly probable.

All the complexity results depend on the expected number of regions defined by the r objects of the sample, and empty (with respect to the sample). As in this article, we focus our interest on cases where the complexity of the result for r objects is $O(r)$, we directly give the results in this case. For more complete results, see the full paper (available from the author).

Lemma 1 *The expected number of conflicts between the regions empty at stage k and the l^{th} object is $O(1)$ (with $k < l$).*

Lemma 2 *The expected number of conflicts between the regions created (and thus empty) at stage k and the l^{th} object is $O\left(\frac{1}{k}\right)$ (with $k < l$).*

Lemma 3 *The expected number of regions created by the insertion of the k^{th} object is $O(1)$.*

The following theorem summarizes the complexity results. The proof is done by summing the quantity in Lemmas 1, 2 and 3 in different manners.

Theorem 4 *If n objects yield $O(n)$ empty regions (all complexities are expected) :*

1. *The size of the conflict graph at stage k is $O(n-k)$.*
2. *$O\left(\frac{n-k}{k}\right)$ edges are created in the conflict graph at stage k. The whole cost of the algorithm is $O\left(\sum_{k=1}^{n} \frac{n-k}{k}\right) = O(n \log n)$.*
3. *The size of the influence graph at stage k is $O(k)$.*
4. *The cost of inserting the l^{th} object in the influence graph is $O(\log l)$. The whole cost of the algorithm is $O\left(\sum_{l=1}^{n} \log l\right) = O(n \log n)$.*
5. *The cost of inserting the l^{th} object in the influence graph knowing the conflicts at stage j is $O\left(\log \frac{l}{j}\right)$.*

Sketch of proof.

1. The size of the conflict graph is its number of edges. At stage k an edge corresponds to a conflict between a region empty at stage k and the l^{th} object with $l > k$. Thus summing quantity of Lemma 1 for $k < l \leq n$ yields the result.
2. An edge of the conflict graph between F and the l^{th} object is created at stage k if F is created at stage k and if F is in conflict with the l^{th} object. Thus summing quantity of Lemma 2 for $k < l \leq n$ yields the result.
3. By the bounded number of sons condition, the size of the influence graph is equal to its number of nodes. This number is simply the sum over all the regions of the probability for a region to be a node of the graph (see Lemma 3).
4. During the insertion of the l^{th} object, the conflicts are located by a traversal of the influence graph. A node F is visited if it is in conflict with the l^{th} object. By summing over $k < l$ the stage of creation of F the quantity of Lemma 2, we get the result.
5. Same result starting the summation at $k = j$. □

4 Accelerated algorithms

The principle of accelerated algorithms, introduced by Seidel [5], speed up the algorithm using Theorem 4.5. The idea is : *if the conflict graph at stage k is known, the insertion in the influence graph can be done faster.* At the beginning, the influence graph is constructed in the usual way, and for some stages N_i, the conflict graph at stage N_i is computed using a direct method exploiting some additional structural information on the objects.

To insert the l^{th} object in the influence graph ($N_i < l \leq N_{i+1}$), the conflicts at stage N_i are found using the conflict graph, and then the conflicts at stage $l-1$ are deduced by traversing the influence graph. By choosing $N_i = \left\lfloor \frac{n}{\log^{(i)} n} \right\rfloor$ (where $\log^{(i)}$ denotes i iterations of log), the cost of inserting objects in the influence graph between the key values N_i and N_{i+1} is

$$\sum_{N_i < j \leq N_{i+1}} O\left(\log \frac{j}{N_i}\right) \leq \sum_{j \leq N_{i+1}} O\left(\log \left[\frac{j}{n} \log^{(i)} n\right]\right)$$
$$\leq N_{i+1} O\left(\log^{(i+1)} n\right) \leq O(n)$$

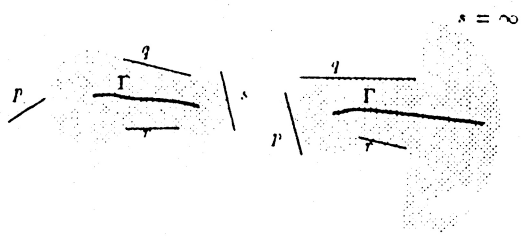

Figure 1: Examples of regions for the EDT

For an efficient application of this principle, it is necessary to be able to determine the conflict graph in a direct way from the whole set of objects, and the set of empty regions of a sample r. We suppose that this can be done in expected time $O(n)$ (remember that the expected size of this graph is $O(n-r)$).

Thus the cost between two key values, for the two steps : the influence graph step, and the direct construction of the conflict graph is $O(n)$. As $N_{(\log^* n)-1} \leq n < N_{\log^* n}$ the number of relevant key values is $\log^* n$ and the whole cost of the algorithm is $O(n \log^* n)$.

Theorem 5 *If Theorem 4 applies and if the conflict graph between the objects and the empty regions of a random sample can be computed in $O(n)$ expected time, then the accelerated algorithm runs in $O(n \log^* n)$ time.*

5 Applications

The first application is the triangulation of a simple polygon. This problem can be solved in linear time by a determinitic algorithm of Chazelle [3], impossible to implement in practice. The solution of Seidel yields a simple randomized algorithm in $O(n \log^* n)$ to compute the vertical visibility map.

This algorithm is not detailed here, the reader can refer to [5]. Seidel's analysis is simpler than that of Section 3 and cannot be generalized directly because he uses special properties of his application.

Influence graphs for Voronoï diagrams

This paragraph shows a randomized algorithm to compute the Voronoï diagram of a set of points or line segments in the plane in $O(n \log n)$ expected time. The next paragraphs will be devoted to accelerated algorithms in $O(n \log^* n)$ for Voronoï diagrams.

We consider here the case of the Voronoï diagram of a set of line segments in the plane, for the usual euclidean distance (the dual is the edge Delaunay triangulation). The Voronoï diagram of a set of points is obviously a special case.

Here, the objects are the line segments of the euclidean plane. A region is defined by four objects p, q, r, s : consider Γ the portion of the bisector of q and r between the two points equidistant of pqr and qrs respectively ; the region $pqrs$ is the union of the disks tangent to q and r whose centers lie on Γ. If $s = \infty$ then Γ extends to infinity, and the disk centered at infinity is an half-plane (see Figure 1). A line segment and a region are in conflict if and only if they intersect, thus a region is empty iff it corresponds to an edge of the Voronoï diagram.

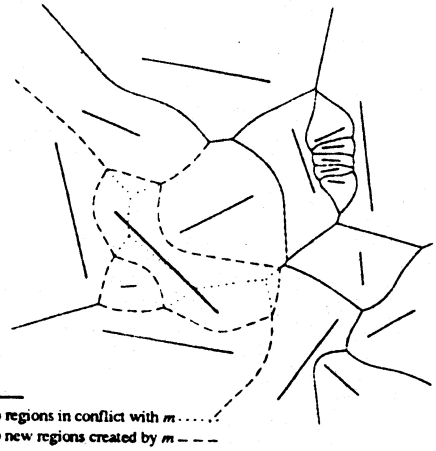

Figure 2: Insertion of m in the Voronoï diagram

When a new segment m is added, the influence graph allows the determination of the empty regions in conflict with m, they correspond to disappearing edges of the Voronoï diagram. Let E be such an edge. If one of the two end-points of E is still a valid vertex of the Voronoï diagram, we compute in constant time the portion of E that remains in the new diagram. The new region associated with that new edge becomes a son of the region associated with E. We then connect the new vertices of the Voronoï diagram (which are new end-points lying on old edges) by edges supported by new bisectors (see Figure 2). The region corresponding to a new edge E' is made son of the regions associated with the unique path of disappearing edges that joins the two end points of E' (the set of disappearing edges form a tree, as can easily be shown).

The size of the Voronoï diagram is $O(n)$, Theorem 4 applies : the Voronoï diagram can be computed in $O(n \log n)$ time using the influence graph or the conflict graph.

Accelerated Delaunay triangulation of points

For a set of points in the plane, it is possible to exploit a partial knowing of the Delaunay triangulation to speed up its complete reconstruction. More precisely, our technique applies if any connected spanning subgraph T of the final Delaunay triangulation with a bounded degree d is known in advance. The Euclidean Minimum Spannig Tree of the points is such a graph. The existence of a determinitic algorithm computing the Delaunay triangulation knowing the MST in $o(n \log n)$ time remains open. Another case happen if the points arer the vertices of a convex polygon. This problem is solved in deterministic way in [1] using a complicated divide and conquer algo-

rithm whose complexity is linear (with a high constant). We show here how our method yields an efficient and simpler algorithm.

The Influence graph algorithm is described above, we have just here to describe the algorithm for a direct determination of the conflict graph at stage k.

First, we show that the expected number of intersection points between T and the Delaunay triangulation of a sample of the points is $O(dn)$. Let vw be an edge of T and ab an edge of the Delaunay triangulation of the sample. There exists an empty region $abcd$ of the sample. If vw intersects ab, then one of the two points v or w lies necessarily in the ball circumscribing abc because if it is not true, a circle passing through v and w must contain either a or b and vw cannot be a Delaunay edge in the final triangulation. So, without lost of generality, suppose that v is in conflict with a region $abcd$, the number of intersection points with ab is bounded by the number of such points v in conflict with $abcd$ multiplied by the maximal degree d of a vertex of T. By summing over all regions, the expected number of intersection points is $O(dn)$.

At this time, it is clearly possible to find the Delaunay triangle in the sample containing each vertex of T by a simple traversal of T and computing all the intersection points. The other conflicts can be deduced using the adjacency relations in the Delaunay triangulation of the sample. Thus Theorem 5 applies : knowing a spanning subgraph of the Delaunay triangulation with maximal degree d, the whole triangulation can be constructed in expected time $O(nd \log^* n)$.

If the points are the vertices of a convex polygon, the edges of this polygon form a spanning subgraph of the Delaunay triangulation where each vertex has degree 2. Using Theorem 5 we conclude that the Delaunay triangulation of a convex polygon can be computed in $O(n \log^* n)$ expected time.

The EMST also verifies the hypothesis, its edges are in the Delaunay triangulation and its maximal degree is less than 6. (Two edges incident to a same vertex must form an angle greater than $\frac{\pi}{3}$.) Thus, knowing the EMST the Delaunay triangulation can be computed in $O(n \log^* n)$ expected time.

Accelerated medial axis of a simple polygon

The influence graph can be used to compute the Voronoï diagram of a set of line segments. If these segments form a simple polygon, or more generally if they are connected then the algorithm can be speed up. The Voronoï diagram of a simple polygon is called its medial axis. The existence of a deterministic $o(n \log n)$ algorithm is open, the convex case is solved in $O(n)$ time [1].

Let the line segments $s_0, \ldots, s_{n-1}$ be a simple polygon, $s_i = p_i p_{i+1}$ ($p_0 = p_n$). For a sample of size k, $s_{\sigma(1)} \ldots s_{\sigma(k)}$, the Voronoï diagram has been already computed. Then we show how to construct the conflict graph in linear time.

From the line segment $s_{\sigma(1)} = p_{\sigma(1)} p_{\sigma(1)+1}$ the regions defined by $p_{\sigma(1)+1}$ are found. Using the adjacency relations in the Voronoï diagram, all the regions in conflict with $s_{\sigma(1)+1} = p_{\sigma(1)+1} p_{\sigma(1)+2}$ are reported and one region containing point $p_{\sigma(1)+2}$ is kept apart to initialize the search for the next line segment $s_{\sigma(1)+2}$. By a single walk around the polygon, the whole conflict graph is computed. The complexity of this algorithm is proportional to the number of conflicts reported, which is expected to be $O(n)$. Using Theorem 5 the medial axis of a simple polygon (or any connected planar graph) can be done in $O(n \log^* n)$ expected time.

6 Conclusion

This paper presents various applications of a general scheme of randomized accelerated algorithms. If a problem can be solved in $O(n \log n)$ time using the usual randomized technique of the conflict graph or the influence graph, it is often possible to use some additional information to speed up the algorithm, and by merging both concepts of the conflict and influence graphs to reach a complexity of $O(n \log^* n)$.

This paradigm is applied in Section 5 to two problems having known deterministic solutions of optimal worst case complexities $\Theta(n)$: the triangulation of a simple polygon, and the Delaunay triangulation of a convex polygon. These optimal algorithms are fairly complicated and if the complexities are not improved here, we propose much more simpler algorithms. For the edge Delaunay triangulation of a simple polygon and the Delaunay triangulation of a set of points knowing the minimum spanning tree, the existence of $O(n \log^* n)$ randomized algorithm, even if no $o(n \log n)$ algorithms was known before, makes us conjecture that the complexity of these problems is $\Theta(n)$. Computing Delaunay knowing the MST in $\Theta(n)$ time will be very interesting because it will prove the equivalence between the two problems (the MST can be deduced from Delaunay in $\Theta(n)$). This technique is powerful and may probably be applied to other problems whose complexity is $\Omega(n)$ and $O(n \log n)$.

References

[1] A. Aggarwal, L. Guibas, J. Saxe, and P. Shor. A linear time algorithm for computing the Voronoï diagram of a convex polygon. *Discr. and Comp. Geom.*, 4:591–604, 1989.

[2] J. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. On-line geometric algorithms with good expected behaviours. In *IMACS*, 1991. Also Tech. Report INRIA 1285, Aug. 1990.

[3] B. Chazelle. Triangulating a simple polygon in linear time. In *IEEE FOCS*, pages 220–230, Oct. 1990.

[4] K. Clarkson and P. Shor. Applications of random sampling in computational geometry, II. *Discr. and Comp. Geom.*, 4(5), 1989.

[5] R. Seidel. *A simple and Fast Randomized Algorithm for Computing Trapezoidal Decompositions and for Triangulating Polygons.* Tech. Report B 90-07, Univ. Berlin, Oct. 1990.

*The author would like to acknowledge Monique Teillaud and Jean-Daniel Boissonnat for a careful reading of the paper.