Randomization yields simple 0(n log n) algorithms for difficult (n) problems

Olivier Devillers

International Journal of Computational Geometry & Applications Vol. 2, No. 1 (1992) 621-635.

RANDOMIZATION YIELDS SIMPLE $O(n \log^\star n)$ ALGORITHMS FOR DIFFICULT $\Omega(n)$ PROBLEMS*

OLIVIER DEVILLERS†
INRIA, 2004 Route des Lucioles, B.P.109
06561 Valbonne cedex (France)

Received April 1991
Revised August 1991
Communicated by Kurt Mehlhorn

ABSTRACT

We use here the results on the influence graph[1] 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 $O(n \log^\star n)$.
This technique applies in the following applications : triangulation of a simple polygon, skeleton of a simple polygon, Delaunay triangulation of points knowing the EMST (euclidean minimum spanning tree).
Keywords: Randomized algorithms, Influence graph, Conflict graph, Skeleton of a polygon, Delaunay triangulation, Euclidean minimum spanning tree

1. Introduction

The classical approach of computational geometry is the search for algorithms having the best possible worst case complexity. Unfortunately, for difficult problems, the algorithms become fairly complicated and the use of sophisticated data structures yields unpractical algorithms. Furthermore, the authors generally study only the order of magnitude of the complexity but too complicated algorithms give implicitely high constants. For example, although the $\Theta(n)$ time triangulation of a simple polygon by Bernard Chazelle[2] is a beautiful theoretical result, it does not yield a practical algorithm for real data on a real computer.

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 made on the data themselves, so this

* This work has been supported in part by the ESPRIT Basic Research Action Nr. 3075 (ALCOM).
† Olivier.Devillers@alcor.inria.fr

approach is different from a classical probabilistic point of view where, for example, the data are supposed to verify a Poisson's distribution.

The two major techniques for incremental randomized constructions use the *conflict* and *influence graphs* respectively. The conflict graph[3,4,5] is a bipartite graph linking the already constructed results to the data not yet inserted. The algorithms using such a structure are obviously static, i.e. the whole set of data must be known in advance to initialize the conflict graph.

The influence graph is an alternative approach[1,6,7]. In this structure, all the intermediate results are linked together to allow the insertion of further data. The data do not need to be known in advance and can be inserted on-line, these algorithms are called semi-dynamic. The analysis is still randomized, but as the data are not known in advance, they cannot be shuffled and must verify the randomization hypothesis. More precisely, the influence graph is a randomized view of an on-line algorithm and not really a randomized algorithm. There are also some recent results[8,9,10] that allow deletions in such structures and obtain fully dynamic algorithms.

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 the case where the segments are connected via their endpoints, Seidel[11] showed that merging the two kinds of graphs results in a speed up of the algorithm. More precisely, the visibility map of a simple n-gon is constructed in $O(n \log^{\star} n)$ expected time, using a simple and practical algorithm.[a]

This paper proposes simpler proofs for the complexity of the conflict and influence graphs that allow to extend Seidel's technique to other applications ; the two main algorithms presented in this paper are the computation of the skeleton of a simple polygon and the Delaunay triangulation of points knowing their euclidean minimum spanning tree. For the two problems the expected complexity is $O(n \log^{\star} n)$, and the algorithms are simple and easy to code. The existence of a $o(n \log n)$ deterministic solution is still open.

2. Conflict and Influence Graphs

2.1. Description of the Problem

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 hyper-planes 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.

[a] $\log^{\star} n = \inf\{k; \log^{(k)} n \leq 1\}$; for $16 < n \leq 65532$, $\log^{\star} n = 4$ and for $65532 < n \leq 2^{65532}$, $\log^{\star} n = 5$. In other words for any reasonable data set, $\log^{\star} n = 4$ and for any imaginable computer where addresses are stored with less than 65532 bits, $\log^{\star} n \leq 5$.

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 deducible 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.

2.2. The Conflict Graph

Clarkson and Shor[3] developed 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 precisely 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.

2.3. 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[1] 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 in such a way that the influence range of a new region is included in the union of the influence ranges of its parents.

2.4. Update Conditions

For the sake of simplicity, we will make some hypotheses. These hypotheses are not really necessary and are relaxed in[1] ; but they are fulfilled by a large class of geometric problems and allow to express the results in a simple way.

- Given a region F and an object O, the test to decide whether or not O is in conflict with F can be performed in constant time.
- If the new object O added to the current set is found to be in conflict with k empty regions then the computation of the new empty regions requires $O(k)$ time.
- In the influence graph, the parents of the new regions can be computed in $O(k)$ time and the number of sons of a node is bounded.
- In the conflict graph, let O be the new object and let k' be the number of conflicts between the empty regions in conflict with O and the objects not in the current set. Then the computation of conflicts between newly created regions and the objects not in the current set can be done in time $O(k')$.

3. Analysis

The classical analysis of these techniques uses random sampling to bound the number of regions in conflict with at most k objects, and deduces time and space bounds for the above algorithms. We propose here a simple analysis where only bounds for the number of empty regions and for the number of regions with a single conflict are needed. In various applications these bounds can be computed directly, without using random sampling techniques.

Let us 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.

We denote :

- ω an event, i.e. one of the $n!$ orders.
- $X_{k,l,F}(\omega)$ is 1 if region F is created by the insertion of the k^{th} object (empty at stage k) and is in conflict with the l^{th} object, 0 otherwise (always 0 if $k \geq l$).
- $Y_{k,l,F}(\omega)$ is 1 if region F is empty at stage k and is in conflict with the l^{th} object, 0 otherwise.
- $X_{k,l}(\omega) = \sum_F X_{k,l,F}(\omega)$ is the number of conflicts between the regions created by the insertion of the k^{th} object and the l^{th} object in the order ω.
- $Y_{k,l}(\omega) = \sum_F Y_{k,l,F}(\omega)$ is the number of conflicts between the regions empty at stage k and the l^{th} object.
- For a random sample of $\mathcal{S}$ of size r, $f_{\mathcal{S}}(r)$ is the expected number of regions defined by the r objects of the sample and empty (with respect to the sample) and $f'_{\mathcal{S}}(r)$ denotes the expected number of regions defined by the r objects of the sample with exactly one conflict (with an object of the sample).

Lemma 1 *The expected value of* $Y_{k,l}$ *(with* $k < l$*) is* $\frac{f'_{\mathcal{S}}(k+1)}{k+1}$.

Proof. Let ω be a given ordering on $\mathcal{S}$. Suppose that the l^{th} object O is introduced immediately after the k first elements. $Y_{k,l}(\omega)$ is the number of regions in conflict with O. By averaging over ω, the k first objects plus O may be any sample of size $k+1$ with the same probability, and O may be any element of the sample with probability $\frac{1}{k+1}$, which yields the result. $\square$.

Lemma 2 *The expected number of regions created by the insertion of the* k^{th} *object is less than* $\frac{bf_{\mathcal{S}}(k)}{k}$.

Proof. Similar to the preceding one. $\square$.

Lemma 3 *The expected value of* $X_{k,l}$ *(with* $k < l$*) is less than* $\frac{b}{k}\frac{f'_{\mathcal{S}}(k+1)}{k+1}$.

Proof. If we know that $Y_{k,l,F}(\omega) = 1$ then $X_{k,l,F}(\omega) = 1$ provided that one of the objects describing F is the k^{th} in the order ω ; this is the case with probability less than $\frac{b}{k}$ since the number of objects defining F is less than b.

We compute the expected value of $X_{k,l}$. The sum is over $\mathcal{F}_{\mathcal{S}}$, the set of regions defined by objects of $\mathcal{S}$:

$$
\begin{aligned}
E(X_{k,l}) &= \sum_{F\in\mathcal{F}_{\mathcal{S}}} E\left(X_{k,l,F}\right) \\
&= \sum_{F\in\mathcal{F}_{\mathcal{S}}} P\left(X_{k,l,F}=1\right) \\
&= \sum_{F\in\mathcal{F}_{\mathcal{S}}} \left[P\left(Y_{k,l,F}=1\right)P\left(X_{k,l,F}=1|Y_{k,l,F}=1\right)\right. \\
&\qquad\qquad \left.+P\left(Y_{k,l,F}=0\right)P\left(X_{k,l,F}=1|Y_{k,l,F}=0\right)\right] \\
&< \sum_{F\in\mathcal{F}_{\mathcal{S}}} E\left(Y_{k,l,F}\right)\frac{b}{k}+0 \\
&= E\left(Y_{k,l}\right)\frac{b}{k} \\
&= \frac{f'_{\mathcal{S}}\left(k+1\right)}{k+1}\frac{b}{k} \qquad \text{using Lemma 1}
\end{aligned}
$$

$\square$.

Theorem 1 *The complexity of the operations on the influence and conflict graphs are the following :*

1. *The expected size of the conflict graph at stage* k *is* $(n-k)\frac{f'_{\mathcal{S}}(k+1)}{k+1}$.
2. *The expected number of edges of the conflict graph created at stage* k *is less than* $\frac{b(n-k)}{k}\frac{f'_{\mathcal{S}}(k+1)}{k+1}$
3. *The expected size of the influence graph at stage* k *is less than* $\sum_{j=0}^{k}\frac{bf_{\mathcal{S}}(j)}{j}$.
4. *The expected cost of inserting the* l^{th} *object in the influence graph is less than* $\sum_{j=0}^{l-1}\frac{b}{j}\frac{f'_{\mathcal{S}}(j+1)}{j+1}$.
5. *The expected cost of inserting the* l^{th} *object in the influence graph knowing the conflicts at stage* k *is less than* $\sum_{j=k}^{l-1}\frac{b}{j}\frac{f'_{\mathcal{S}}(j+1)}{j+1}$.

Proof.

1 The size of the conflict graph is its number of edges. At stage k the regions present in the conflict graph are exactly the empty regions at stage k, the number of edges reaching the j^{th} object is $Y_{k,j}$, thus the whole size of the conflict graph is $E\left(\sum_{j=k+1}^{n} Y_{k,j}\right)$.

2 An edge of the conflict graph between F and the j^{th} object is created at stage k if F is created at stage k and if F is in conflict with the j^{th} object. By summing over j we get $E\left(\sum_{j=k+1}^{n} X_{k,j}\right)$.

3 By the bounded number of sons conditions, 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. By Lemma 2 the expected number of nodes created at stage j is less than $\frac{b f_{\mathcal{S}}(j)}{j}$.

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 the stage of creation j of F we get $E\left(\sum_{j=1}^{l-1} X_{j,l}\right)$. According to update conditions, the number of visited nodes in the influence graph is linearly related to the cost of the insertion.

5 Same result starting the summation at $j = k$. $\square$.

In the applications described in Section 5 $f_{\mathcal{S}}$ and $f'_{\mathcal{S}}$ are both linear. In such a case, the complexities get a more explicit expression stated in the following theorem. Furthermore, if a direct expression of $f'_{\mathcal{S}}$ is not available, it is possible to show[3] that $f'_{\mathcal{S}}(r) = O\left(f_{\mathcal{S}}\left(\lfloor \frac{r}{2} \rfloor\right)\right)$, so it is enough to suppose that $f_{\mathcal{S}}$ is linear.

Theorem 2 *If $f_{\mathcal{S}}(r) = O(r)$,*

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

Proof. This theorem is simply a corollary of Theorem 1. In Point 2, the whole cost can be deduced because the update conditions ensure that the cost of the algorithm is related to the total number of structural changes in the conflict graph. $\square$.

4. Accelerated Algorithms

The principle of accelerated algorithms, introduced by Seidel,[11] is to exploit Theorem 2 Point 5 in order to achieve a speed up. 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.

If $f_{\mathcal{S}}(r)$ is supposed to be $O(r)$, by choosing $N_i = \left\lfloor \frac{n}{\log^{(i)} n} \right\rfloor$ (where $\log^{(i)}$ denotes i iterations of log), the expected cost of inserting objects in the influence graph between the key values N_i and N_{i+1} is

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

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 expected 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 $N_{(\log^\star n)-1} \leq n < N_{\log^\star n}$ the number of relevant key values is $\log^\star n$ and the whole expected cost of the algorithm is $O(n \log^\star n)$.

Theorem 3 *If $f_{\mathcal{S}}(r) = O(r)$, 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 $O(n \log^\star n)$ expected time.*

5. Applications

5.1. Triangulating a Simple Polygon

The first application is the triangulation of a simple polygon. This problem can be solved in linear time by a deterministic algorithm of Chazelle,[2] impossible to implement in practice. Seidel's solution yields a simple randomized algorithm in $O(n \log^\star n)$ to compute the vertical visibility map.

This algorithm is not detailed here, the reader can refer to Seidel's paper.[11] Seidel's analysis is simpler than that of Section 3 and cannot be generalized directly because he uses special properties of his application. More precisely, in Seidel's algorithm the nodes of the influence graph visited during the insertion of a new object form a single path. This fact is used in Seidel's analysis, and yields directly the value of Y : $\forall \omega, Y_{k,l}(\omega) = 1$.

5.2. Influence and Conflict Graphs for Voronoï Diagrams

This section presents 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 sections will be devoted to accelerated algorithms in $O(n \log^\star n)$ for special 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 of this diagram is called edge Delaunay triangulation : EDT). The Voronoï diagram of a set of points is obviously a particular case and is solved by this algorithm (a detailed description of this algorithm for points in any dimensions can be found in a previous paper[12]).

We first recall the definition of the Voronoï diagram. $\mathcal{S}$ is a set of objects, here points or line segments in the euclidean plane $\mathcal{E}$. We define the Voronoï cell $V(p)$ of $p \in \mathcal{S}$ as $V(p) = \bigcap_{q \in \mathcal{S} \setminus \{p\}} \{m \in \mathcal{E}; \delta(p,m) \leq \delta(q,m)\}$ where δ denotes the euclidean distance.

The Voronoï diagram $Vor_{\mathcal{S}}$ is the union of the Voronoï cells of each object of $\mathcal{S}$, see Figure 1 for an example. These cells intersect only on their boundaries and form

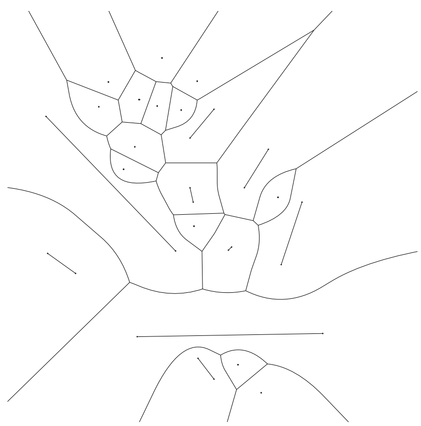


Figure 1: Example of Voronoï diagram

a partition of the plane. An important property of the Voronoï diagram is that, since each edge is a portion of a bisecting line, the maximal empty disk centered on a Voronoï vertex touches three objects and the maximal empty disk centered on a Voronoï edge touches two objects.

A randomized incremental construction can solve efficiently the problem of computing $Vor_{\mathcal{S}}$. The first point is the definition of objects, regions and conflicts, such that the Voronoï diagram is characterized by the empty regions. The objects are naturally the line segments (or the points). As said above, an edge Γ of a the Voronoï diagram is a part of a bisecting line of p and q, and the endpoints of Γ are equidistant to pqr and pqs ; such a Voronoï edge (defined by four segments) is called a region and denoted (pq, r, s). Another segment m is said to be in conflict with (pq, r, s) if m intersects the union of the maximal empty disks centered along Γ. In other words, m is in conflict with (pq, r, s) if Γ is not an edge of $Vor_{\{p,q,r,s,m\}}$, see Figure 2.

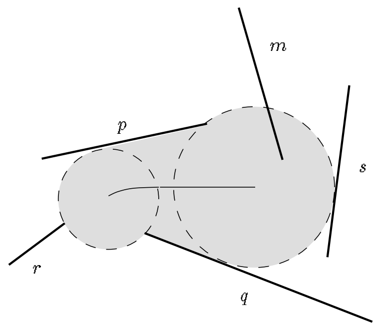


Figure 2: m is in conflict with region (pq,r,s)

In fact it is necessary to be a little more precise in the definition of regions to hold on some special cases. Firstly, to describe the unbounded edges of a Voronoï diagram, we just use a new symbol : ∞. The region (pq, r, ∞) corresponds to an unbounded part of the bisecting line of p and q, see Figure 3. Secondly, to ensure

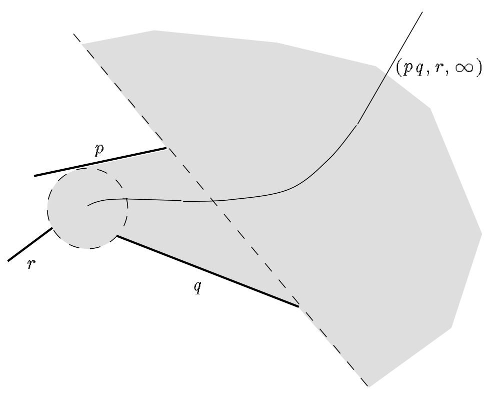


Figure 3: An unbounded region (pq,r,∞)

the connectivity of the Voronoï diagram, it is necessary to add some "virtual" edges to "bound" the unbounded Voronoï cells, see Figures 4 and 5. Thirdly, in some special cases, the notation (pq, r, s) may be ambiguous, but (pq, r, s) can define at most two Voronoï edges. In case of ambiguity, the two regions are distinguished by the notations $(pq, r, s)^+$ and $(pq, r, s)^-$, see Figure 5.

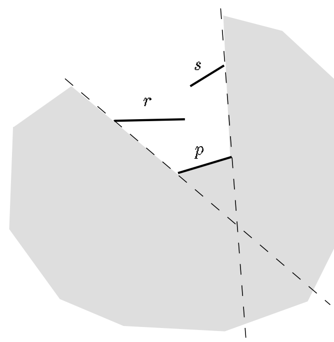


Figure 4: Another kind of unbounded region $(p\infty, r, s)$

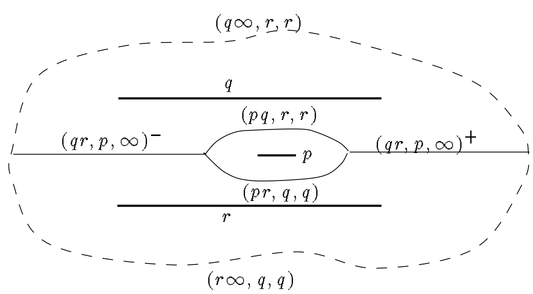


Figure 5: Region (qr, p, ∞) can be ambiguous

It is easy to see that with these definitions, a region is empty if and only if it corresponds to an edge of the Voronoï diagram.

The second aspect of the design of a randomized incremental algorithm is the description of the update procedure for the influence or conflict graphs. If 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. Consider now (pq, r, s) as a conflicting region. Possibly one (or even two) portions of the corresponding edge remain in the new diagram, then the new edge (pq, r, m) for example is made son of (pq, r, s). So, look at the disappearing part of the Voronoï diagram (see Figure 6), it is a tree whose leaves are the vertices of $V(m)$ the new Voronoï cell, they are also the new endpoints of the shortened Voronoï edges described above. Consider a new Voronoï edge on the boundary of $V(m)$ and let x and y be its endpoints. There exists a unique path of disappearing edges (they form a tree) linking x and y. The new empty region corresponding to the new edge is made son of all conflicting regions corresponding to edges on this path. In such a way, an old edge is traversed by two paths (one for each side of the edge). A region in conflict with m has at most four sons, two corresponding to edges on the boundary of $V(m)$ and possibly two shortened edges (see for example Figure 5 and suppose p is inserted last).

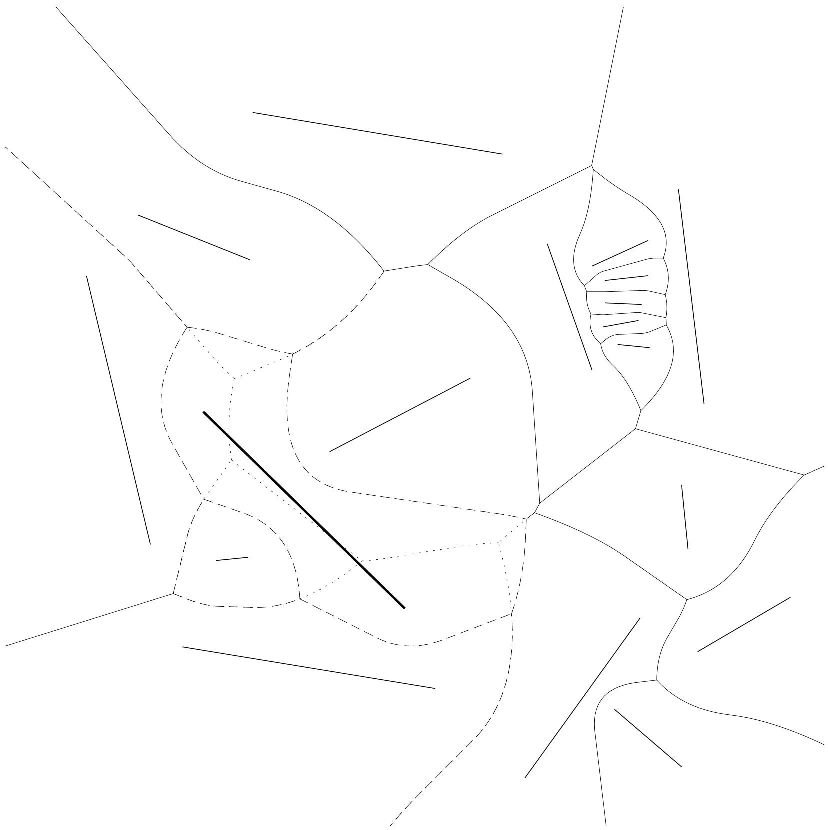
For the conflict graph technique, the conflicts with m are directly available and the conflicts of a disappearing region must be distributed among at most four new regions.

The update conditions are verified, thus, to apply the complexity results, we just need to know $f_{\mathcal{S}}$ and $f'_{\mathcal{S}}$. Here $f_{\mathcal{S}}(n) = O(n)$ because this quantity is related to the size of the order 1 Voronoï diagram[13] and $f'_{\mathcal{S}}(n) = O(n)$ is related to the size of the order 2 Voronoï diagram.[13] The result of Theorem 2 applies : the Voronoï diagram (or the edge Delaunay triangulation) can be computed in $O(n \log n)$ time using the influence graph (or the conflict graph).

5.3. Accelerated Delaunay Triangulation Knowing the Euclidean Minimum Spanning Tree

It is possible to use the Euclidean Minimum Spanning Tree of a set of points (EMST) to speed up the construction of the Delaunay triangulation. The existence of a deterministic algorithm solving this problem in $o(n \log n)$ time remains open. In fact, our technique applies not only for the EMST, but for any connected spanning subgraph T of the Delaunay triangulation with bounded degree d. All the edges of T are edges of the final Delaunay triangulation.

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 otherwise, 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 loss of generality, suppose that v is in conflict



The new segment m is in bold line.
The dotted edges correspond to regions in conflict with m.
The dashed edges correspond to new regions created by m.

Figure 6: Insertion of m in the Voronoï diagram

with a region $abcd$. The number of intersection points between T and 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, for each vertex of T, the Delaunay triangle in the sample containing the vertex 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 3 applies : knowing a spanning subgraph of the Delaunay triangulation with maximal degree d, the whole triangulation can be constructed in expected time $O(nd \log^\star n)$.

The EMST verifies the hypothesis, its edges are in the Delaunay triangulation[14] and its maximal degree is less than 6. (Two edges incident to the same vertex must form an angle greater than $\frac{\pi}{3}$.) Thus, knowing the EMST the Delaunay triangulation can be computed in $O(n \log^\star n)$ expected time.

Remark : if the points are vertices of a convex polygon, then this polygon is a correct spanning graph T of degree 2, thus the Delaunay triangulation of a convex polygon can be computed in $O(n \log^\star n)$ expected time. This problem is solved deterministically by Aggarwal et al.[15] using a complicated divide and conquer algorithm whose complexity is linear (with a high constant). There exists also a still unpublished algorithm by Paul Chew[16] whose randomized expected complexity is linear. The idea is to remove the points from the convex hull in the reverse insertion order, only maintaining the current convex hull. Thus when inserting the point again one conflicting region is known (namely the infinite one) and the search for conflicts is avoided.

5.4. Accelerated Skeleton of a Simple Polygon

The influence graph can be used to compute the Voronoï diagram of a set of line segments (also called the skeleton). If these segments form a simple polygon, or more generally if they are connected then the algorithm can be speed up.

The existence of a deterministic algorithm with complexity $o(n \log n)$ has not already been settled. Aggarwal et al.[15] provides an $O(n)$ deterministic algorithm for a convex polygon, and Chew's idea[16] applies also in the special case of a convex polygon.

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 3 the skeleton of a simple polygon (or any connected planar graph) can be computed in $O(n \log^\star n)$ 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 ; by merging both concepts of the conflict and influence graphs a complexity of $O(n \log^\star n)$ can be achieved.

This paradigm is applied in Section 5 to two problems having known deterministic solutions of optimal worst case complexities $\Theta(n)$, but these algorithms are fairly complicated. These problems are the triangulation of a simple polygon,[2] and the Delaunay triangulation of a convex polygon.[15] In these cases, previous bounds are not improved, but the randomized algorithms are much simpler.

For the two others applications, no $o(n \log n)$ algorithm was known before. These problems are the edge Delaunay triangulation of a simple polygon and the Delaunay triangulation of a set of points knowing the euclidean minimum spanning tree. Computing the Delaunay triangulation knowing the EMST in $\Theta(n)$ time will be very interesting because it will prove the equivalence between the two problems (the EMST can be deduced from the Delaunay triangulation in $\Theta(n)$ time).

This technique is powerful and may probably be applied to other problems whose complexity is $\Omega(n)$ and $O(n \log n)$.

Acknowledgements

The author would like to thank the anonymous referee and other people who communicated to him Chew's algorithm,[16] Monique Teillaud and Jean-Daniel Boissonnat for a careful reading of the paper and Jean-Pierre Merlet for supplying him with his interactive drawing preparation system JPdraw .

References

1. J.D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete and Computational Geometry*. To be published. Available as Technical Report INRIA 1285. Abstract published in IMACS 91 in Dublin.
2. B. Chazelle. Triangulating a simple polygon in linear time. In *IEEE Symposium on Foundations of Computer Science*, pages 220–230, October 1990.
3. K.L. Clarkson and P.W. Shor. Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry*, 4(5), 1989.
4. K. Mulmuley. On levels in arrangements and Voronoï diagrams. *Discrete and Computational Geometry*, 6:307–338, 1991.
5. K. Mehlhorn, S. Meiser, and C. Ó'Dúnlaing. On the construction of abstract Voronoï diagrams. *Discrete and Computational Geometry*, 6:211–224, 1991.

6. J.D. Boissonnat, O. Devillers, and M. Teillaud. A semi-dynamic construction of higher order Voronoï diagrams and its randomized analysis. *Algorithmica.* To be published. Available as Technical Report INRIA 1207. Abstract published in Second Canadian Conference on Computational Geometry 1990 in Ottawa.

7. L.J. Guibas, D.E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoï diagrams. *Algorithmica.* To be published. Abstract published in LNCS 443 (ICALP 90).

8. O. Devillers, S. Meiser, and M. Teillaud. Fully dynamic Delaunay triangulation in logarithmic expected time per operation. *Computational Geometry Theory and Applications.* To be published. Available as Technical Report INRIA 1349. Abstract published in LNCS 519 (WADS91).

9. K.L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. June 1991. Manuscript.

10. O. Schwarzkopf. Dynamic maintenance of geometric structure made easy. In *IEEE Symposium on Foundations of Computer Science*, October 1991. Full paper available as Technical Report B 91-05 Universität Berlin.

11. R. Seidel. A simple and fast randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry Theory and Applications*, 1, 1991.

12. J.D. Boissonnat and M. Teillaud. On the randomized construction of the Delaunay tree. *Theoretical Computer Science.* To be published. Available as Technical Report INRIA 1140.

13. D.T. Lee. On k-nearest neighbor Voronoï diagrams in the plane. *IEEE Transactions on Computers*, C-31:478–487, 1982.

14. F.P. Preparata and M.I. Shamos. *Computational Geometry : an Introduction.* Springer-Verlag, 1985.

15. A. Aggarwal, L.J. Guibas, J. Saxe, and P.W. Shor. A linear time algorithm for computing the Voronoï diagram of a convex polygon. *Discrete and Computational Geometry*, 4:591–604, 1989.

16. P. Chew. A simple randomized linear time algorithm for computing the Voronoï diagram of a convex polygon. Unpublished.