A Semi-Dynamic Construction of Higher Order Voronoï Diagrams and its Randomized Analysis

Jean-Daniel Boissonnat[†] **Olivier Devillers**[†] **Monique Teillaud**[†]

Key-words : Computational Geometry, On-line Algorithms, Average Case Analysis, Random Sampling, Conflict Graph, Influence Graph.

Published in *Algorithmica* **9:329-356, 1993**.

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

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

Abstract

The k-Delaunay tree extends the Delaunay tree introduced in [1, 2]. It is a hierarchical data structure that allows the semi-dynamic construction of the higher order Voronoï diagrams of a finite set of n points in any dimension. In this paper, we prove that a randomized construction of the k-Delaunay tree, and thus, of all the order $\leq k$ Voronoï diagrams, can be done in $O(n \log n + k^3 n)$ expected time and $O(k^2 n)$ expected storage in the plane, which is asymptotically optimal for fixed k. Our algorithm extends to d dimensional space with expected time complexity $O\left(k^{\lceil \frac{d+1}{2} \rceil + 1} n^{\lfloor \frac{d+1}{2} \rfloor}\right)$ and space complexity $O\left(k^{\lceil \frac{d+1}{2} \rceil} n^{\lfloor \frac{d+1}{2} \rfloor}\right)$. The algorithm is simple and experimental results are given.

Résumé

Le k-arbre de Delaunay est une généralisation de l'arbre de Delaunay introduit dans [1, 2]. C'est une structure hiérarchique qui permet de construire dynamiquement les diagrammes de Voronoï d'ordres supérieurs d'un ensemble de n points en dimension quelconque. Dans cet article, nous prouvons qu'une construction randomisée du k-arbre de Delaunay, et donc de tous les diagrammes de Voronoï d'ordre $\leq k$, peut être effectuée en temps moyen $O(n \log n + k^3 n)$ et mémoire $O(k^2 n)$ dans le plan, ce qui est asymptotiquement optimal pour k fixé. L'algorithme se généralise en dimension d, avec une complexité moyenne de $O\left(k^{\lceil \frac{d+1}{2} \rceil + 1} n^{\lfloor \frac{d+1}{2} \rfloor}\right)$ et une place mémoire $O\left(k^{\lceil \frac{d+1}{2} \rceil} n^{\lfloor \frac{d+1}{2} \rfloor}\right)$. L'algorithme est simple et des résultats expérimentaux sont présentés.

1 Introduction

The order k Voronoï diagram has been introduced in [3] in order to deal with k-closest points and related distance relationships. Lee [4] gave the first algorithm for constructing the order k Voronoï diagram of a set of n points (or sites) in the plane. This algorithm constructs the order k Voronoï diagram from the order $(k-1)$ Voronoï diagram in time $O(kn \log n)$. Thus the order k Voronoï diagram (in fact, the family of all order j Voronoï diagrams for $1 \leq j \leq k$ — the order $\leq k$ Voronoï diagrams for short) can be constructed in time $O(k^2 n \log n)$. This bound can be tightened to $O(n \log n + k^2 n)$ using the result of [5].

Chazelle and Edelsbrunner [6] developed two versions of an algorithm that is better for large values of k. The first one takes $O(n^2 \log n + k(n-k) \log^2 n)$ time and $O(k(n-k))$ storage while the other takes $O(n^2 + k(n-k) \log^2 n)$ time and $O(n^2)$ storage.

A radically different approach, pioneered by Clarkson [7], uses random sampling. Clarkson's algorithm determines the order k Voronoï diagram of n sites in the plane in time $O(kn^{1+\varepsilon})$ with a constant factor that depends on ε.

More recently, in order to gain simplicity, several authors have designed algorithms which are incremental and randomized. Such an approach has been applied successfully for constructing Voronoï diagrams in the plane [8, 9] and in d-space [2, 10, 11]. A common point to all these randomized algorithms is that no distribution assumptions are made as it is the case, for example, in [12]. Hence the results remain valid for any set of points, provided that the points are inserted at random.

The algorithms in [10, 9, 11] are incremental in the sense that the points are introduced one at a time. But all the points need to be known in advance and maintained in an auxiliary data structure, the so called conflict graph.

The well known relationship between higher order Voronoï diagrams in d dimensions and arrangements of hyperplanes in $d+1$ dimensions can be used for the design of an algorithm that constructs the order $\leq n-1$ Voronoï diagrams in time and storage $O(n^{d+1})$, as shown by Edelsbrunner, O'Rourke and Seidel [13].

In $d > 2$ dimensions, Clarkson [10] has shown that the size of the order $\leq k$ Voronoï diagrams is $O(k^{\lceil \frac{d+1}{2} \rceil} n^{\lfloor \frac{d+1}{2} \rfloor})$. Recently, Mulmuley [11, 14] has obtained a randomized algorithm whose expected complexity meets this bound for $d > 2$ and whose complexity is $O(nk^2 + n \log n)$ for $d = 2$. This algorithm also uses a conflict graph.

None of the previous algorithms, except [2, 8] are semi-dynamic. If one new site is to be added, the Voronoï diagram has to be entirely reconstructed.

In this paper, we present an algorithm that is semi-dynamic. After each insertion of a new site, the algorithm updates a data structure, called the k-Delaunay tree. This structure generalizes the Delaunay tree, introduced in [1, 2] to compute the Delaunay triangulation (and, by duality, the Voronoï diagram) of a set of points. The k-Delaunay tree contains all the successive versions of the order $\leq k$ Voronoï diagrams and allows fast point location.

As any semi-dynamic algorithm constructing the Voronoï diagram, ours cannot be very good in the worst-case. However, a randomized analysis shows that it is efficient on the average. The analysis of our algorithm has some similarity with the ones in

[2, 10, 8]. Our main result states that if we randomize the insertion sequence of the n sites, the k-Delaunay tree (and thus the order $\leq k$ Voronoï diagrams) can be constructed in expected time $O(n \log n + k^3 n)$ in two dimensions and expected storage $O(k^2 n)$.

Our algorithm extends to higher dimensions. For a given value d of the dimension, its expected time complexity is $O\left(k^{\lceil \frac{d+1}{2} \rceil + 1} n^{\lfloor \frac{d+1}{2} \rfloor}\right)$ and its expected space complexity is $O\left(k^{\lceil \frac{d+1}{2} \rceil} n^{\lfloor \frac{d+1}{2} \rfloor}\right)$.

The overall organization of the paper is the following. In Section 2, we define the k-Delaunay tree in two dimensions and present an algorithm for its construction. In Section 3, we analyse the complexity of the randomized construction of the k-Delaunay tree and thus, of all order $\leq k$ Voronoï diagrams. Section 4 shows how to use the k-Delaunay tree for searching the l nearest neighbours of a given point. In Section 5, we extend our results to d dimensions. Last but not least, Section 6 presents experimental results which provide evidence that the algorithm is very effective in practice for small values of k.

2 The k-Delaunay tree in two dimensions

2.1 The order k Voronoï Diagram

2.1.1 Definition and properties

Let $\mathcal{E}$ denote the euclidean plane. Let $\mathcal{S}$ be a set of n sites (points of $\mathcal{E}$). We will assume that no four sites lie on a same circle, and that no three sites are collinear. For $\mathcal{T}$ a subset of points of $\mathcal{S}$, the generalized Voronoï polygon of $\mathcal{T}$ is defined by

$$V(\mathcal{T}) = \{p, \forall v \in \mathcal{T}, \forall w \in \mathcal{S} \setminus \mathcal{T}, \delta(p, v) < \delta(p, w)\}$$

where δ denotes the euclidean distance in $\mathcal{E}$. $V(\mathcal{T})$ is the locus of points p such that p is closer to each point of $\mathcal{T}$ than to any point not in $\mathcal{T}$. The order k Voronoï diagram is defined as

$$Vor_k(\mathcal{S}) = \{V(\mathcal{T}),\, \mathcal{T} \subseteq \mathcal{S},\, |\mathcal{T}| = k\}$$

Let p_1, p_2, p_3 be three sites of $\mathcal{S}$, v the center of the circle passing through p_1, p_2, p_3, and $B(p_1, p_2, p_3)$ the open disk bounded by it. Since no four sites are cocircular, this circle passes through no other site, and v is the intersection of the three bisecting lines of $[p_1, p_2]$, $[p_2, p_3]$ and $[p_1, p_3]$. The disk $B(p_1, p_2, p_3)$ contains some sites, and R will denote the set of all these sites. If $|R| = k$, then v is a vertex of $Vor_{k+1}(\mathcal{S})$ and $Vor_{k+2}(\mathcal{S})$ (see for example [15]) :

- v is the common point of $V(R \cup \{p_1\})$, $V(R \cup \{p_2\})$ and $V(R \cup \{p_3\})$ in $Vor_{k+1}(\mathcal{S})$. v is called a *close-type vertex* of this diagram (see Figure 1).

- v is the common point of $V(R \cup \{p_1, p_2\})$, $V(R \cup \{p_2, p_3\})$ and $V(R \cup \{p_3, p_1\})$ in $Vor_{k+2}(\mathcal{S})$. v is a *far-type vertex* of this diagram (see Figure 2).

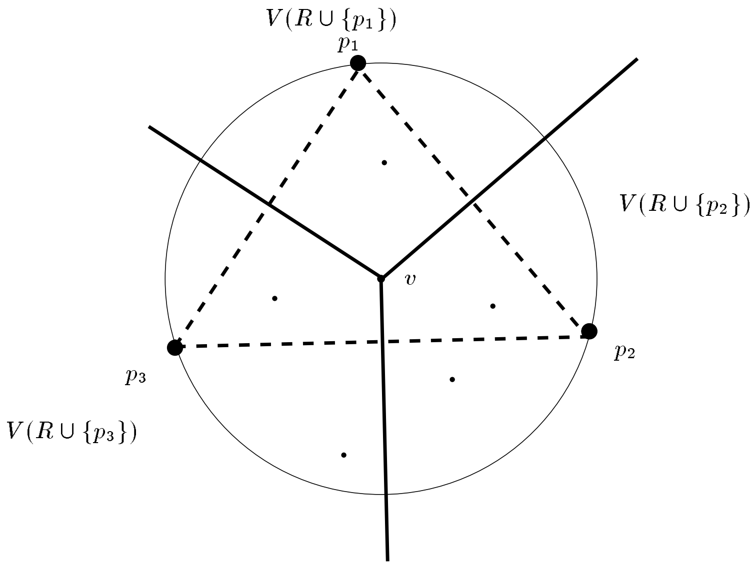


Figure 1: A close-type vertex in $Vor_{k+1}(\mathcal{S})$

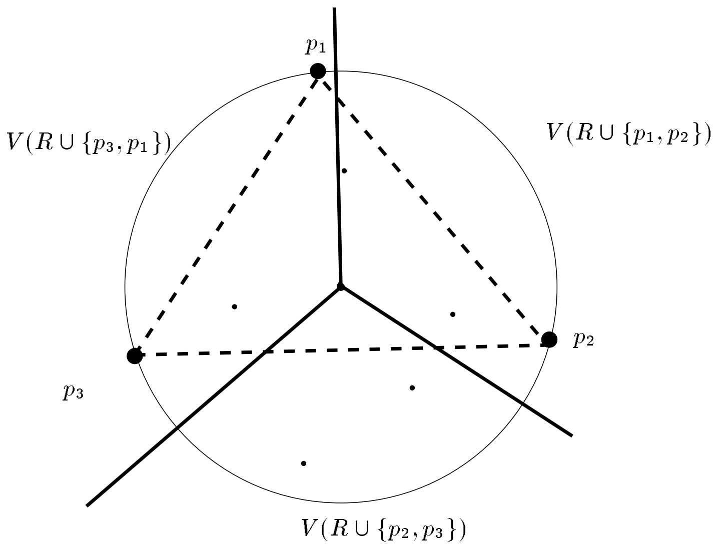


Figure 2: A far-type vertex in $Vor_{k+2}(\mathcal{S})$

Summarizing, any triangle T whose vertices are sites of $\mathcal{S}$ and whose circumscribing disk contains k sites in its interior corresponds (is *dual*) to a vertex of both $Vor_{k+1}(\mathcal{S})$ and $Vor_{k+2}(\mathcal{S})$.

Reciprocally, a vertex of $Vor_k(\mathcal{S})$ is dual to a triangle whose circumscribing disk contains $k-1$ or $k-2$ sites in its interior.

We end this section with a lemma which will be useful in the sequel.

Lemma 2.1 *Let $\mathcal{T} \subset \mathcal{S}$. The Voronoï polygon $V(\mathcal{T})$ does not change if we add to $\mathcal{T}$ and $\mathcal{S}$ some new points lying in the convex hull of $\mathcal{T}$. More precisely $V(\mathcal{T})$ in $Vor_{|\mathcal{T}|}(\mathcal{S})$ is equal to $V(\mathcal{T} \cup \mathcal{R})$ in $Vor_{|\mathcal{T}\cup\mathcal{R}|}(\mathcal{S} \cup \mathcal{R})$ if the points of $\mathcal{R}$ lie in the interior of the convex hull of $\mathcal{T}$.*

Proof : Let $\mathcal{R}$ be a set of points not in $\mathcal{S}$, and lying in the interior of the convex hull of $\mathcal{T} \subset \mathcal{S}$. We denote $V_{\mathcal{S}}(\mathcal{T})$ the Voronoï polygon of $\mathcal{T}$, where $\mathcal{T}$ is considered as a subset of $\mathcal{S}$ (i.e. in $Vor_{|\mathcal{T}|}(\mathcal{S})$). We first prove that $V_{\mathcal{S}}(\mathcal{T}) \subset \mathcal{V}_{\mathcal{S}\cup\mathcal{R}}(\mathcal{T} \cup \mathcal{R})$.

Let $m \in V_{\mathcal{S}}(\mathcal{T})$, $p \in \mathcal{T} \cup \mathcal{R}$, and $q \in (\mathcal{S} \cup \mathcal{R}) \setminus (\mathcal{T} \cup \mathcal{R}) = \mathcal{S} \setminus \mathcal{T}$.

- if $p \in \mathcal{T}$, then $\delta(m,p) < \delta(m,q)$, since $m \in V_{\mathcal{S}}(\mathcal{T})$.
- if $p \in \mathcal{R}$, then $p = \sum_i \alpha_i t_i$, where $t_i \in \mathcal{T}$ and $\alpha_i \in I\!R^+, \sum_i \alpha_i = 1$.

$$\begin{aligned}
\delta(m,p) &\leq \sum_i \alpha_i \delta(m,t_i) \\
&\qquad \text{since } \delta \text{ is associated to the euclidean norm,} \\
&\qquad \text{and thus } x \mapsto \delta(m,x) \text{ is a convex function} \\
&< \sum_i \alpha_i \delta(m,q) \text{ since } t_i \in \mathcal{T} \\
&= \delta(m,q)
\end{aligned}$$

In both cases, $\delta(m,p) < \delta(m,q)$, from which we deduce that :
$m \in V_{\mathcal{S}\cup\mathcal{R}}(\mathcal{T} \cup \mathcal{R})$.

The reciprocal inclusion is straightforward, which achieves the proof. $\square$

2.1.2 Including and excluding neighbours

In the sequel, we will denote $B(T)$ the interior of the disk circumscribing triangle T.

For a triangle T, we will define 2 neighbours through each of its 3 edges : one will be called the *including* neighbour and the other one the *excluding* neighbour. This notion of neighbourhood corresponds to the actual notion of adjacency in the higher order Voronoï diagrams.

Let E be an edge of T and p be the third vertex of T. Let us consider a moving disk B whose boundary passes through the end points of E, and whose center moves along the bisecting line of E. Starting from $B = B(T)$, we can move B in two opposite directions : the one such that $p \in B$ is called the including direction and the other such that $p \notin B$ is called the excluding direction. We stop moving B as soon as its boundary encounters a site different from the end points of E. Let q_i (*resp.* q_e) be the first site encountered in the including (*resp.* excluding) direction. The triangle T_i (*resp.* T_e) having E as an edge and q_i (*resp.* q_e) as a vertex will be called the *including neighbour* (*resp. excluding neighbour*) of T through edge E (see Figure 3). Notice that q_i and q_e may be on either side of E.

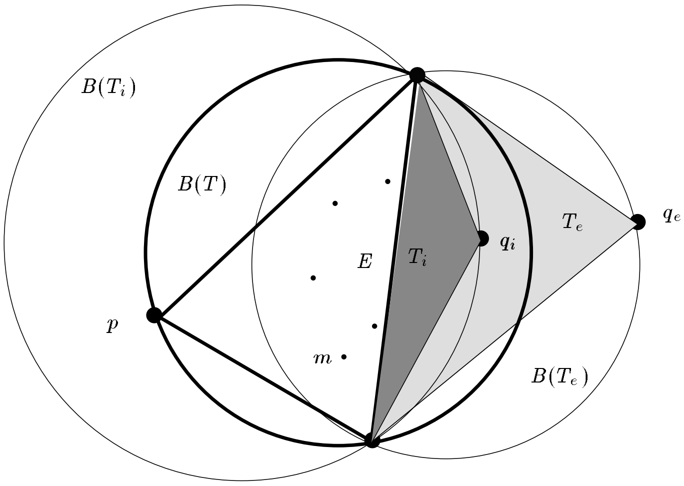


Figure 3: Including and excluding neighbours

Remark 2.1 In the sequel, we will also speak of the neighbours of a triangle T in the direction including a site m in $B(T)$ and in the direction excluding m. The neighbour of T through E in the direction excluding m (*resp.* including m) is the excluding (*resp.* including) neighbour of T if m and p are on the same side of E and the including (*resp.* excluding) neighbour of T otherwise.

Remark 2.2 If S is the excluding (*resp.* including) neighbour through E for T, then T is a neighbour of S, but T may be either the including neighbour through E of S or the excluding one, depending on which side of E lies q_e (*resp.* q_i). The neighbourhood relationships are not reciprocal.

Remark 2.3 The following property will be useful in the sequel :

$$B(T) \subset B(T_i) \cup B(T_e)$$

Hence, if a site m lies into $B(T)$, we can deduce that m lies into either $B(T_i)$ or $B(T_e)$.

If $B(T)$ contains k sites, then $B(T_i)$ contains $k+1$ sites if q_i lies outside $B(T)$ (the k sites in $B(T)$ plus p), or k sites if q_i lies inside $B(T)$ (the k sites in $B(T)$ plus p minus q_i). In a similar way, $B(T_e)$ contains k sites if q_e lies outside $B(T)$ (the k sites in $B(T)$), or $k-1$ sites if q_e lies inside $B(T)$ (the k sites in $B(T)$ minus q_e).

Speaking in terms of Voronoï vertices, if $B(T)$ contains k sites, T is dual to a *close-type* vertex t of Vor_{k+1} and its excluding neighbours are dual to the adjacent vertices of t in Vor_{k+1}. Similarly, T is dual to a *far-type* vertex t' of Vor_{k+2}, and its including neighbours are dual to the adjacent vertices of t' in Vor_{k+2}.

2.1.3 A semi-dynamic algorithm for constructing the order $\leq k$ Voronoï diagrams

Our algorithm is based on the well known semi-dynamic algorithm of [16] for constructing the Delaunay triangulation. Each site is introduced, one after another, in each of the order $\leq k$ Voronoï diagrams and each diagram is subsequently updated. We will describe this algorithm at the same time as the k-Delaunay tree, in the following sections, but it is actually independent of that data structure.

2.2 Construction of the k-Delaunay tree

The k-Delaunay tree is a hierarchical structure that we use to construct the order $\leq k$ Voronoï diagrams. The skeleton of the k-Delaunay tree is the Delaunay tree presented in [1] and [2]. It is not really a tree but a rooted direct acyclic graph. As in the Delaunay tree, the nodes are associated to triangles. We will often use the same word *triangle* for both a triangle and its associated node.

Following our algorithm, each site is introduced one after another and we keep all triangles in a hierarchical manner in the k-Delaunay tree, creating appropriate links between "old" (i.e. created before the introduction of the site) and "new" triangles (i.e. created after the introduction of the site). This will allow to efficiently locate a new site in the current structure.

We define the *current width* of a triangle T dual to a vertex of some higher order Voronoï diagram to be the number of already inserted sites lying inside $B(T)$.

2.2.1 Initialization

For the initialization step we choose 3 sites. They generate one finite triangle and six half planes (considered as infinite triangles) limited by the supporting lines of the finite triangle. The 4 triangles of current width zero will be the sons of the root of the tree. Their neighbourhood relationships in the Delaunay triangulation are created. The 3 triangles of current width 1 are linked to the preceding ones by their neighbourhood relationships in the order 2 Voronoï diagram.

2.2.2 Inserting a new site

The k-Delaunay tree will be constructed so that it satisfies the following property :

($\mathcal{P}$) *all the triangles of current width strictly less than k are present in the k-Delaunay tree.*

Hence, the triangles dual to the vertices of the order $\leq k$ Voronoï diagrams are all present in the structure. Moreover, we keep their adjacency relationships in the corresponding Voronoï diagrams. The k-Delaunay tree thus contains the whole information necessary to construct all the order $\leq k$ Voronoï diagrams.

Let us suppose that the k-Delaunay tree has been constructed for the already inserted sites and satisfies the above property ($\mathcal{P}$). Let m be the next site to be inserted.

Let S be a triangle dual to a vertex of some of the order $\leq k$ Voronoï diagrams, to be created after the insertion of m in order to satisfy ($\mathcal{P}$). S is called a *new triangle*. S has m as a vertex ; let E be the edge of S not containing m. S has an including neighbour T through E, and an excluding neighbour R through E. It is plain to observe that R and T were necessarily neighbours before the insertion of m and that $B(T)$ contains m while $B(R)$ does not. Figure 4 shows the four typical situations.

Conversely, let T be a triangle dual to a vertex of some of the order $\leq k$ Voronoï diagrams, whose disk $B(T)$ contains m. Let E be an edge of T, and p be the third vertex of T. If the neighbour R of T through E in the direction excluding m does not contain m in its disk, then we will create a new triangle S by linking m to E. T becomes the including neighbour of S through E and R its excluding one.

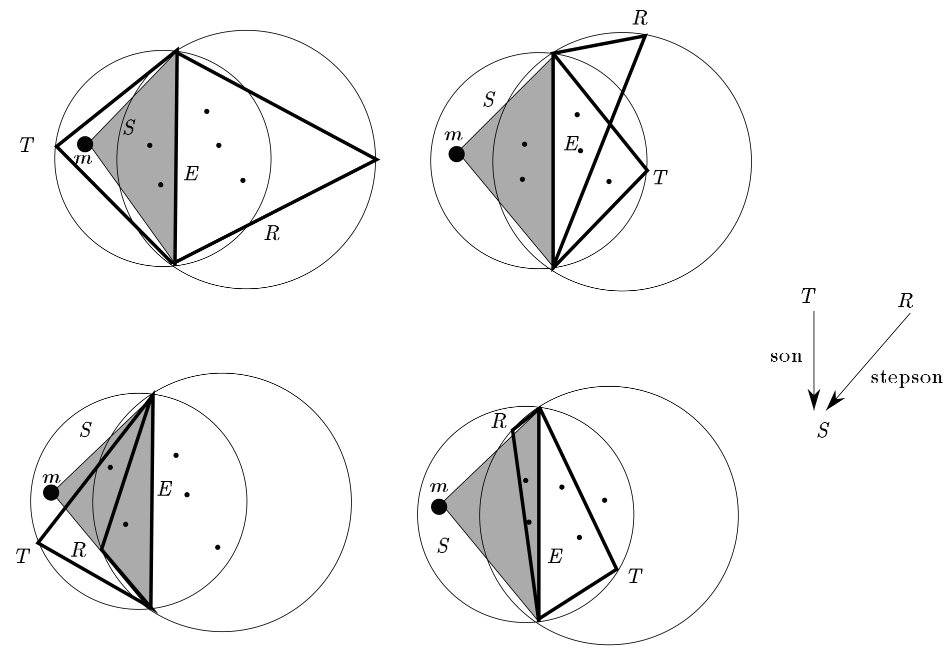
If now the current width of S is l , the current width of T, before the insertion of m, was l or $l-1$ and the current width of R is still l or $l-1$, as before the insertion of m (see Figure 4). Thus property ($\mathcal{P}$) implies that T and R were associated to nodes in the k-Delaunay tree before the insertion of m, the current width of T is increased by one and is now $l+1$ or l. If this number is larger or equal to k, then T is marked as *full* (T is no longer dual to a vertex of some order $\leq k$ Voronoï diagram).

If the current width of S is zero when it is created, we then update the tree by making S a *son* of T and a *stepson* of R.

> *Remark 2.4* We can thus notice that the subgraph of the k-Delaunay tree obtained by recursively traversing all links to sons and stepsons, starting from the root, is the Delaunay tree (the 1-Delaunay tree) presented in [1] : a triangle can be created in the (1-)Delaunay tree only if its current width is zero ; at this moment, it is the neighbour of its stepfather, and the current width of its father is incremented, so it becomes at once full. There are no additional neighbourhood relationships involving triangles of current width larger than zero, because the (1-)Delaunay tree only deals with neighbourhood relationships between Delaunay triangles.

We also update the neighbourhood relationships between the triangles which are not marked as full.

In order to ensure that ($\mathcal{P}$) still holds, it is sufficient to apply the above procedure to *all* the triangles T whose circumscribing disks contain m. Thus we need to find all those triangles. This will be performed by Procedure `location` while the creation



If l is the current width of the new triangle S, the current width of T before the insertion of m, the current width of T after the insertion and the current width of R are respectively :

- l, $l+1$ and l in the upper-left case
- $l-1$, l and l in the upper-right case
- l, $l+1$ and $l-1$ in the bottom-left case
- $l-1$, l and $l-1$ in the bottom-right case.

Figure 4: Inserting a new site

of the new triangles and the maintenance of the neighbourhood relationships will be performed by Procedure creation. These two procedures are detailed below.

When a node receives sons, its width increases, and it can only get new sons when its width is zero, so the total number of sons of a node is at most three. Notice however that the number of stepsons is unbounded ; unfortunately, all stepsons are useful as shown in [2].

Nevertheless, we have the following property :

Lemma 2.2 *The total number of links to sons and stepsons in the k-Delaunay tree is less than twice the number of nodes.*

> *Proof :* Each newly created node (the 7 first ones excepted) has exactly one father and one stepfather. □

2.2.3 Structure of the k-Delaunay tree

As previously noted, the k-Delaunay tree is a directed acyclic graph. Each node of the k-Delaunay tree is associated to a triangle. The node associated to triangle T contains the following informations :

1. three links to the three sites which are vertices of T
2. the center and the radius (more exactly the squared radius) of $B(T)$, which can also be used to mark T as finite or infinite
3. its at most 3 sons
4. the list of its stepsons
5. the last site located in T
6. the last site propagated in T (these last two fields will be used by Procedure location)
7. the current width of T, which can also be used to mark T as full
8. the six neighbours of T

The first six fields are the same as in the (1-)Delaunay tree (see Remark 2.4). In the (1-)Delaunay tree, as soon as the current width of a triangle T is equal to 1, it becomes full, so there is only a mark remembering whether T is full or not ; another difference is that there are only three neighbours (they are excluding neighbours), that are the neighbours of T in the Delaunay triangulation, if T is not full.

2.2.4 Procedure location

If m belongs to the circumscribing disk of a triangle, we know (see Remark 2.3) that it belongs to the union of the circumscribing disks of its father and of its stepfather. So we will be able to find all the triangles whose disks contain m by traversing the k-Delaunay tree.

In fact, we do not need to really traverse the k-Delaunay tree. It is sufficient to traverse only the Delaunay tree, which is a subgraph of the k-Delaunay tree (see Remark 2.4 and the previous section), until we find one Delaunay triangle in conflict with m. Then we follow the neighbourhood relationships to find all triangles, of current width strictly less than k, in conflict with m. We store them in a list $T(m)$, the edges of these triangles are the possible candidates to be used for the creation of new triangles.

Procedure location is described in Figure 5.

```
Initialize the list T(m) as the empty list
location(m,root of the k-Delaunay tree)

Procedure location(m,node)
    (* The node is associated to triangle T *)
        if the last site located in T is not m and
        if m lies into B(T), then
            m becomes the last site located in T;
            for each son, location(m,son);
            for each stepson, location(m,stepson);
            if the current width of T is 0, then
                propagate(m,T);
                stop Procedure location.

Procedure propagate(m,T)
    m becomes the last site propagated in T;
    add T to the list T(m);
    increment the current width of T;
    if the current width of T is now k, then mark T as full;
    for each neighbour N of T
        if the last site propagated in N is not m and
        if N is not full, then
            if m lies into B(N), then propagate(m,N).
```

Figure 5: Locating a site in the k-Delaunay tree

2.2.5 Procedure creation

We go through the list $T(m)$. Let T be a triangle of this list and E one of its edges. If the neighbour R of T through E in the direction excluding m does not contain m, we then create S by linking m to E, and the son and stepson relations involving S, if S's current width is 0. Procedure creation is described in Figure 6.

Procedure creation($T(m)$)

for each triangle T of $T(m)$
    for each edge E of T
        if the neighbour R of T through E in the direction
            excluding m does not contain m in its disk, then
        • create the triangle S having vertex m and edge E,
            and its associated node;
        • declare R as the excluding neighbour of S through E
            and update the reciprocal relation;
        • declare T as the including neighbour of S through E
            and update the reciprocal relation;
        • if the current width of S is 0, then
            create the relations : S son of T and S stepson of R;
create the neighbourhood relationships between the new triangles.

Figure 6: Creating the new triangles

Let us now see how we can maintain the neighbourhood relationships between triangles.

As already mentioned, when S is created, R is the excluding neighbour of S through edge E common to T and R, and T is the including one. We can easily compute the reciprocal relations : R and T were neighbours before the insertion of m (notice that R may be either an excluding or an including neighbour of T ; remember also that the relations between neighbours are not symmetric, see Remark 2.2). If T was the excluding (*resp.* including) neighbour of R through E, then now S is the excluding (*resp.* including) neighbour of R. Similarly, if R was the excluding (*resp.* including) neighbour of T through E, then now S is the excluding (*resp.* including) neighbour of T.

In order to create the neighbourhood relationships between the new triangles, we proceed as follows. Let us suppose that the insertion of m creates $x(m)$ new triangles $T_1, T_2, \ldots, T_{x(m)}$, where $T_i = (m, s_i^0, s_i^1), i = 1, 2, \ldots, x(m)$. The T_i's are dual to vertices of various $Vor_l(\mathcal{S}), l \leq k$. The adjacency relationships must be created in each $Vor_l(\mathcal{S}), l \leq k$. Our goal is to find, for each T_i $(i = 1, \ldots, x(m))$, its excluding and its including neighbours through both edges. To this aim, for each

$s_i^j, i = 1, 2, \ldots, x(m), j = 0, 1$, we sort the list of new triangles having s_i^j as a vertex, according to the abscissa of their center on the bisecting line of $[m, s_i^j]$. By definition, (m, s_i^j, v) and (m, s_i^j, v') are neighbours through $[m, s_i^j]$ *iff* they are successive in this order. We thus only have to go through the list of sorted triangles to obtain the neighbourhood relationships through the corresponding edge.

For each edge (m, s_i^j), the complexity of this sorting is $O(n_i^j \log n_i^j)$, where n_i^j denotes the number of new triangles having (m, s_i^j) as an edge, which is bounded by $2k$. The whole complexity of creating the neighbourhood relationships is thus $O(x(m) \log k)$ for each new site m.

> *Remark 2.5* The $\log k$ appearing in this complexity could be dropped, but it would complicate our algorithm. As we shall see in the sequel, the cost of Procedure creation is dominated by the cost of Procedure location, therefore we do not elaborate on improving its complexity.

3 Analysis of the randomized construction

The above algorithm allows to insert new sites in a dynamic way. In this section, we will prove that this simple algorithm is efficient, for any data set, as long as we randomize the insertion sequence of the sites. Thus, as in [8], our analysis implicitly assumes that all the sites are known in advance (while the algorithm does not need that). However, for an "on-line" application, the following theorem remains valid provided that all sequences have the same probability. So a sufficient (but not necessary) condition for Theorem 3.1 to be valid during an on-line execution is that the sites are generated independently.

This section proves the main result of this paper, stated in the following theorem :

Theorem 3.1 *For any set of n sites, if we randomize the sequence of their insertion, the k-Delaunay tree (and thus the order $\leq k$ Voronoï diagrams) of the n sites can be constructed in expected time $O(n \log n + k^3 n)$ in two dimensions, using expected storage $O(k^2 n)$.*

The remaining of Section 3 is devoted to the proof of Theorem 3.1. Section 3.3 analyses the expected space used to store the k-Delaunay tree, Section 3.4 the expected cost of locating the n sites and Section 3.5 the cost of constructing the successive triangles and their neighbourhood relationships.

3.1 Some definitions

Let X, Y, Z, T be four sites. As in [2] we define the *bicycle* $X(YZ)T$ to be the figure drawn by $B(XYZ)$ and $B(YZT)$ (see Figure 7 which represents one of the possible configurations of a bicycle).

As [8], we define the *width* of triangle XYZ to be the number of sites of the whole set $\mathcal{S}$ contained in the interior of its circumscribing disk. The *width* of a bicycle $X(YZ)T$ will be the number of sites belonging to $B(XYZ)$ and $B(YZT)$, where we do not take

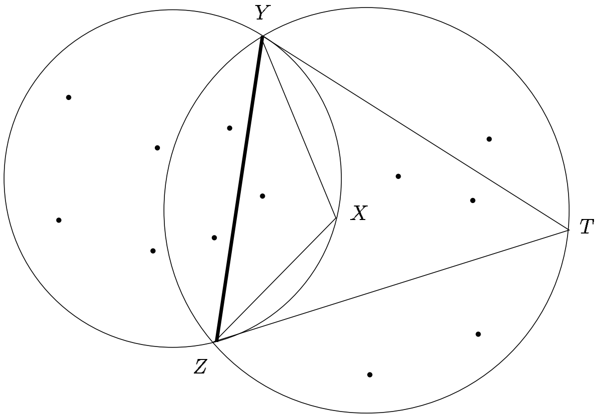


Figure 7: The bicycle $X(YZ)T$

X, Y, Z and T into account. T_j (*resp.* C_j) will denote the set of triangles (*resp.* bicycles) having width j and $T_{\leq j}$ (*resp.* $C_{\leq j}$) the set of triangles (*resp.* bicycles) of width at most j. Recall that the *current width* of a triangle has been defined above as its width at the current stage of the construction ; the *current width* of a bicycle can be defined similarly.

> *Remark 3.1* The width of a triangle (*resp.* of a bicycle) and its current width at some stage of the construction must not be mistaken : the width does not depend on the number of already inserted sites. It depends only on the whole set of sites.

3.2 Results on triangles and bicycles

Lemma 3.1 *Let XYZ be a triangle having width j. XYZ will arise as a vertex of some order $\leq k$ Voronoï diagram during the construction with probability*

$$\begin{cases} \dfrac{k(k+1)(k+2)}{(j+1)(j+2)(j+3)} & \textit{if} \quad j \geq k \\ 1 & \textit{if} \quad j < k \end{cases}$$

> *Proof :* Let l be the current width of XYZ when XYZ is created. The property holds if and only if one of the three sites X, Y and Z is introduced after the l sites inside $B(XYZ)$ (in any order, hence $3(l+2)!$ possible orders for the $l+3$ first sites, and $\begin{pmatrix} j \\ l \end{pmatrix}$ possible sets of l sites among the j ones), and before the $j-l$ remaining sites (which may also be introduced in any order, that is $(j-l)!$ possibilities). There are $(j+3)!$ permutations on

the $j+3$ sites. Thus XYZ of width j appears with current width l with probability :

$$\frac{3\binom{j}{l}(l+2)!\,(j-l)!}{(j+3)!} = \frac{3(l+1)(l+2)}{(j+1)(j+2)(j+3)}$$

The required probability is the sum of the last ones, for all l :

- if $j \geq k$, l can only be $< k$, so we get

$$\sum_{l=0}^{k-1} \frac{3(l+1)(l+2)}{(j+1)(j+2)(j+3)} = \frac{k(k+1)(k+2)}{(j+1)(j+2)(j+3)}$$

- if $j < k$

$$\begin{aligned}\sum_{l\leq j} \frac{3(l+1)(l+2)}{(j+1)(j+2)(j+3)} &= \frac{(j+1)(j+2)(j+3)}{(j+1)(j+2)(j+3)} \\ &= 1\end{aligned}$$

which agrees with Property $(\mathcal{P})$: a triangle with width $< k$ will necessarily appear at some stage of the construction. $\square$

Lemma 3.2 *Let $X(YZ)T$ be a bicycle of width j such that YZT is a son or a stepson of XYZ. The probability that such a bicycle appears during the construction is*

$$\frac{3!}{(j+1)(j+2)(j+3)(j+4)}$$

Proof : A bicycle $X(YZ)T$ is always created with current width 0. Now, to get YZT as a son or a stepson of XYZ, we need the following condition : T must be inserted after X, Y and Z ; thus there are 3! possibilities to insert X, Y, Z and T. Then the j sites inside the bicycle can be inserted in any of the $j!$ possible orders. So the probability is :

$$\frac{3!j!}{(j+4)!} = \frac{3!}{(j+1)(j+2)(j+3)(j+4)}$$

This result also agrees with property $(\mathcal{P})$: a bicycle with width 0 will always appear, and the last site inserted among X, Y, Z and T is T with probability $\frac{1}{4}$. $\square$

The following lemma is a direct consequence of the bound on the size of higher order Voronoï diagrams.

Lemma 3.3 *The number of triangles having width j is*

$$|T_j| \leq (2j+1)n$$

Proof : $|T_j|$ is exactly the number of close-type vertices of the order $j+1$ Voronoï diagram. The result follows from [4]. □

The next lemma is given in [10]. We propose an alternative proof here.

Lemma 3.4 *The number of bicycles having width at most j is*

$$|C_{\leq j}| = O(n(j+1)^3)$$

Proof : We first notice that a given segment joining two points of $\mathcal{S}$ is an edge of at most $2j+2$ triangles of width less than j.

We then bound the number of bicycles of width $\leq j$ by subdividing a bicycle into two adjacent triangles, one of width l and the other of width less than j.

$$\begin{aligned} |C_{\leq j}| &\leq \sum_{l=0}^{j} |T_l| 3(2j+2) \\ &\leq 6(j+1) \sum_{l=0}^{j} O(n(l+1)) \\ &\leq 6(j+1) O(n(j+1)^2) \\ &= O(n(j+1)^3) \end{aligned}$$

□

3.3 Analysis of the expected space used by the k-Delaunay tree

Lemma 3.5 *The expected number of nodes in the k-Delaunay tree is $O(k^2 n)$.*

Proof : This number is the number of all successive vertices arising in all order $\leq k$ Voronoï diagrams during the construction. It is less than

$$\begin{aligned} &\sum_{j=0}^{n-3} \sum_{S \in T_j} Prob(S \text{ arises during the construction}) \\ &= \sum_{j=0}^{k-1} \sum_{S \in T_j} 1 + \sum_{j=k}^{n-3} \sum_{S \in T_j} \frac{k(k+1)(k+2)}{(j+1)(j+2)(j+3)} \text{ (using Lemma 3.1)} \end{aligned}$$

$$= \sum_{j=0}^{k-1} |T_j| + k(k+1)(k+2) \sum_{j=k}^{n-3} \frac{|T_j|}{(j+1)(j+2)(j+3)}$$
$$= O(k^2 n) \text{ , using Lemma 3.3 and } \sum_{j=k}^{n-3} \frac{1}{(j+2)(j+3)} = O(\frac{1}{k})$$

□

We already know that the total number of links to sons and stepsons is less than the number of nodes (Lemma 2.2), and that each node has at most 6 neighbours. We conclude :

Proposition 3.1 *The expected space complexity of the k-Delaunay tree is $O(k^2 n)$.*

3.4 Analysis of the expected cost of Procedure location

We want here to count the number of nodes visited during the construction. During the insertion of m, Procedure location looks at first for a Delaunay triangle in conflict with m. To this end, a triangle YZT may be visited if YZT is the son or the stepson of a triangle XYZ such that $m \in B(XYZ)$. So a bicycle $X(YZ)T$ of width j obliges to visit up to j nodes. Secondly, Procedure propagate is called, and a triangle is visited if it is the neighbour of a triangle XYZ such that $m \in B(XYZ)$, and XYZ is not full. So a triangle of width j obliges to visit up to $6\inf(j, k-1)$ nodes.

Lemma 3.6 *The expected total number of visited nodes during the construction is $O(n \log n + k^3 n)$.*

Proof : The total number of visited nodes during the search for a triangle of the Delaunay triangulation in conflict with a new site is inferior to :

$$\sum_{j=0}^{n-4} \sum_{X(YZ)T \in C_j} j \, Prob \begin{pmatrix} YZT \text{ arises as a son} \\ \text{or a stepson of } XYZ \\ \text{during the construction} \end{pmatrix}$$
$$\leq \sum_{j=0}^{n-4} \sum_{X(YZ)T \in C_j} j \frac{3!}{(j+1)(j+2)(j+3)(j+4)}$$
$$\text{(using Lemma 3.2)}$$
$$\leq \sum_{j=1}^{n-4} \frac{3! j |C_j|}{(j+1)(j+2)(j+3)(j+4)}$$
$$= 3! \sum_{j=1}^{n-4} \frac{j(|C_{\leq j}| - |C_{\leq j-1}|)}{(j+1)(j+2)(j+3)(j+4)}$$
$$= 3! \sum_{j=1}^{n-4} |C_{\leq j}| \left(\frac{j}{(j+1)\dots(j+4)} - \frac{j+1}{(j+2)\dots(j+5)} \right)$$

$$
\begin{aligned}
&= 3! \sum_{j=1}^{n-4} \frac{|C_{\leq j}|(3j-1)}{(j+1)\dots(j+5)} \\
&= O(n \log n) \text{ , using Lemma 3.4.}
\end{aligned}
$$

When such a triangle has been found, Procedure `propagate` then explores all triangles in conflict with the site, using the neighbourhood relationships. The total number of triangles visited during the propagation is less than :

$$\sum_{j=0}^{n-3} \sum_{S \in T_j} 6 \inf(j, k-1)\, Prob\,(S \text{ arises during the construction})$$

Similar calculations as above prove that thais is $O(k^3 n)$, using Lemma 3.1. □

Since it takes constant time to process a node, Lemma 3.6 yields the following proposition :

Proposition 3.2 *The expected cost of Procedure* `location` *is* $O(n \log n + k^3 n)$.

3.5 Analysis of the expected cost of Procedure `creation`

This cost is the cost of creating all the vertices of the order $\leq k$ Voronoï diagrams, plus the cost of maintaining the adjacency relationships, after each insertion of a new site m. If $x(m)$ is the number of new triangles created after the insertion of m, the first cost is $O(x(m))$, since creating a triangle costs a constant time. As we have seen in Section 2.2.5, the second cost is $O(x(m) \log k)$. The overall cost is thus, using Lemma 3.5 :

$$O\left(\sum_m x(m)\right) + O\left(\sum_m x(m) \log k\right) \leq O(k^2 n \log k)$$

As mentioned in Remark 2.5, this could be improved to $O(k^2 n)$ complexity.

Proposition 3.3 *The expected cost of Procedure* `creation` *is* $O(k^2 n \log k)$.

Altogether, Propositions 3.1, 3.2 and 3.3 prove Theorem 3.1.

4 l-nearest neighbours

The k-Delaunay tree contains the combinatorial structure of any order l Voronoï diagram ($l \leq k$). We show in the following section how such a diagram can be extracted from the k-Delaunay tree.

As shown by the analysis of Procedure `location`, the k-Delaunay tree is an efficient data structure to perform point location. Section 4.2 shows that it can be readily used to search the l nearest neighbours of a given point.

4.1 Deducing the order l Voronoï diagram from the k-Delaunay tree ($l \leq k$)

Theorem 4.1 *$Vor_l(\mathcal{S})$ ($l \leq k$) can be deduced from the k-Delaunay tree in time proportional to the size of $Vor_l(\mathcal{S})$, which is $O(ln)$.*

> *Proof :* Remember that the k Delaunay tree maintains all the adjacency relations in $Vor_l(\mathcal{S})$. So we can find $Vor_l(\mathcal{S})$ in time proportional to its size, as soon as we know one of its vertices. We thus only have to find one vertex of each Voronoï diagram.
>
> Assume we know an infinite triangle $pq\infty$ of current width 0. Its finite edge $[pq]$ is necessarily an edge of the current convex hull. Let s_0 be the site such that pqs_0 is a neighbour of $pq\infty$ and that $B(pqs_0)$ is empty. Let $s_1, \ldots, s_{k-1}$ be the sites such that triangle pqs_i is the including neighbour of triangle pqs_{i-1}, $i = 1, \ldots, k-1$. Such sites always exist (provided that $k < n-1$) since $[pq]$ is an edge of the convex hull of the already inserted sites. Each triangle pqs_i is associated to a node in the k-Delaunay tree and its current width is i. Triangle pqs_i is dual to a vertex v_i of $Vor_{i+1}(\mathcal{S})$.
>
> If we maintain a pointer to an infinite triangle of current width 0 (which can be done in constant time at each stage of the construction), we can find the s_i, $i \leq l$, and thus a vertex of each $\leq l$ Voronoï diagrams, in time $O(l)$. $\square$
>
> *Remark 4.1* Now, suppose we want to label the regions of $Vor_i(\mathcal{S})$, $i \leq k$, by the i nearest sites of an (arbitrary) point of the interior of that region. We remark that the label of one region of $Vor_{i+1}(\mathcal{S})$ incident to v_i is $\{s_0, s_1, \ldots, s_i\}$. As soon as we know the label of one particular region, we can deduce the labels of all the other regions by traversing the diagram. Each time we cross an edge e, we come out of one region, say c, and enter another region, say c'. The labels of c and c' differ by only one site. The site of the label of c which is not in the label of c', and the site of the label of c' which is not in the label of c, are precisely the two sites whose bisector supports e. We have to substitute this first site by the second one in the label of c to get the label of c'.

4.2 Finding the l nearest neighbours

Let us consider now the problem of finding the l nearest neighbours ($l \leq k$) of a given point. This problem is equivalent to that of finding the label of the region $V(\{p_1, p_2, \ldots, p_l\})$ of $Vor_l(\mathcal{S})$ which contains point m. It remains to show how to find the label of a region. This point must be clarified since our structure represents vertices of the Voronoï regions and we know, from Lemma 2.1, that the label of a Voronoï polygon is not included in the union of the labels of its vertices and thus, cannot be deduced from them (the label of a vertex consists of the three sites which are the vertices of its dual triangle ; it is also the symmetric difference of the labels of its incident regions).

However, we can take advantage of the fact that we know all the order $\leq l$ Voronoï diagrams to compute the label of a region. The following lemma will help in that task.

Lemma 4.1 *Let $m \in V(\{p_1, p_2, \ldots, p_l\})$ in $Vor_l(\mathcal{S})$, where $p_1, p_2, \ldots, p_l$ are some sites of $\mathcal{S}$. Without loss of generality, we assume that $\delta(p_i, m) \leq \delta(p_j, m)$ if and only if $i \leq j$. Then, for each $i \in \{1, \ldots, l\}$, there exists a vertex v_i of $Vor_i(\mathcal{S})$, such that v_i is dual to a triangle S_i having p_i as a vertex, and such that $m \in B(S_i)$.*

> *Proof :* Since p_l is a l^{th} nearest site from m, we have $\delta(p_i, m) \leq \delta(p_l, m)$, $\forall i = 1, \ldots, l$.
>
> Let q be a point on the line (mp_l) : $q = tm + (1-t)p_l, t \in I\!R$.
>
> $$\begin{aligned} \delta(p_l, q) &= |t|\delta(p_l, m) \\ \delta(m, q) &= |1-t|\delta(p_l, m) \end{aligned}$$
>
> Let us suppose that $t \geq 1$, i.e. m lies between q and p_l.
>
> The following inequalities hold, for $i = 1, \ldots, l$:
>
> $$\begin{aligned} \delta(p_i, q) &\leq \delta(p_i, m) + \delta(m, q) \\ &\leq \delta(p_l, m) + (t-1)\delta(p_l, m) \\ &= t\delta(p_l, m) \\ &= \delta(p_l, q) \end{aligned}$$
>
> So, if we move q on the half line $D^+_{mp_l} = \{tm + (1-t)p_l, t \geq 1\}$ supported by (mp_l), a furthest site from q among $\{p_1, p_2, \ldots, p_l\}$ remains p_l, the same as from m. We can deduce that the intersection of $D^+_{mp_l}$ with the boundary of $V(\{p_1, p_2, \ldots, p_l\})$ is a point q belonging to the bisecting line $Bis(p_l, p')$ of $[p_l, p']$, where p' is a site of $\mathcal{S}$ not in $\{p_1, p_2, \ldots, p_l\}$. Let us denote F the edge of $V(\{p_1, p_2, \ldots, p_l\})$ supported by $Bis(p_l, p')$ (see Figure 8).
>
> Let us define $R = \{r \in Bis(p_l, p') / m \in C(r, p_l)\}$ where $C(r, p_l)$ denotes the disk of center r and radius $\delta(r, p_l)$. R is a half line of $Bis(p_l, p')$ containing q. So $F \cap R \neq \emptyset$, which implies that one of the two end points of F, say v_l, belongs to R. v_l is a vertex of $Vor_l(\mathcal{S})$ dual to a triangle S_l having p_l as one of its vertices, and whose disk contains m.
>
> The lemma is thus proved for p_l. For the other $p_i, i = 1, 2, \ldots, l-1$, the same proof still holds, considering $Vor_i(\mathcal{S})$ instead of $Vor_l\mathcal{S})$. □

If we want to find the label of the region of $Vor_l(\mathcal{S}), l \leq k$, which contains m, we can locate it, by applying Procedure location to the l-Delaunay tree. We first find all the triangles whose balls contain m. We then have to find, among the vertices of those triangles, the l nearest sites from m, which can be done in a straightforward way.

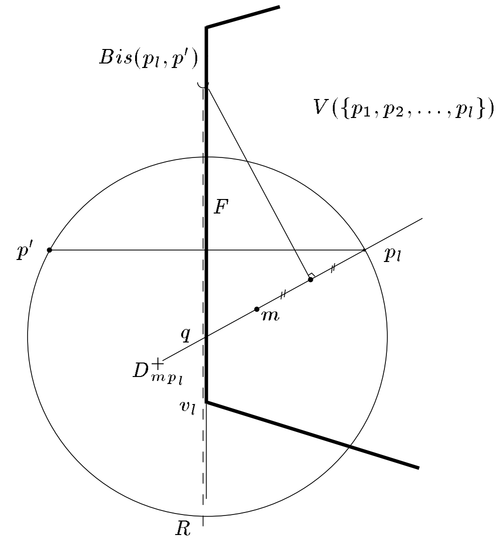


Figure 8: For the proof of Lemma

5 The k-Delaunay tree in higher dimensions

5.1 The order k Voronoï diagram

In this section, we generalize the previous results to higher dimensions. Let $\mathcal{S}$ be a set of n sites in d-space such that no subset of $d+2$ sites lie on a same sphere and no subset of $d+1$ sites are coplanar.

The order k Voronoï diagram in d dimensions is made of cells of dimensions $0, \dots, d$. The vertices of the diagram are dual to d-simplices whose vertices are sites of $\mathcal{S}$. Let $p_1, p_2, \dots, p_{d+1}$ be $d+1$ sites of $\mathcal{S}$ and let $B(\{p_1, \dots, p_{d+1}\})$ be the ball passing through these points, v its center and $\mathcal{R}$ the subset of sites of $\mathcal{S}$ contained in $B(\{p_1, \dots, p_{d+1}\})$. If $|\mathcal{R}| = l$ (i.e. $p_1, p_2, \dots, p_{d+1}$ is a simplex of width l), v is a vertex of all higher order Voronoï diagrams $Vor_k(\mathcal{S})$ for $l+1 \le k \le l+d$.

- v is the common point of the $V(\mathcal{R} \cup \{\sqrt{}\rangle\})$ for $i \in \{1, \dots, d+1\}$ in $Vor_{l+1}(\mathcal{S})$: as in 2 dimensions, we call v a *close-type vertex.*
- In $Vor_{l+h}(\mathcal{S})$, v is the common point of the regions $V(\mathcal{R} \cup \mathcal{P})$ where P may be any subset of size h of $\{p_1, \dots, p_{d+1}\}$; v is incident to $\binom{d+1}{h}$ regions. If $1 < h < d$ we call v a *medium-type vertex.*
- v is the common point of the $V(\mathcal{R} \cup \{\sqrt{}\infty, \dots, \sqrt{}\lceil+\infty\} \setminus \{\sqrt{}\rangle\})$ for $i \in \{1, \dots, d+1\}$ in $Vor_{l+d}(\mathcal{S})$: as in 2 dimensions, we call v a *far-type vertex.*

Summarizing, a close-type vertex of a higher order Voronoï diagram remains in d Voronoï diagrams of successive orders. More generally, a h-face of a higher order Voronoï diagram remains in $d-h$ Voronoï diagrams of successive orders.

The notion of including and excluding neighbours can be generalized. A simplex has $d+1$ excluding neighbours and $d+1$ including ones, one through each of its hyper-faces. To any pair of adjacent simplices correspond an edge in some higher order Voronoï diagram. If v is a close-type vertex (*resp.* far-type vertex) of $Vor_k(\mathcal{S})$, the edges of $Vor_k(\mathcal{S})$ issued from v correspond to the neighbourhood relationships between the simplex dual to v and its excluding (*resp.* including) neighbours. If v is a medium-type vertex, the edges of $Vor_k(\mathcal{S})$ issued from v correspond to both types of neighborhood relationships.

5.2 The d dimensional k-Delaunay tree

We can generalize the structure developed in Section 2.2. The d-dimensional Delaunay tree is a direct acyclic graph satisfying the following property :

> $(\mathcal{P})$ *all the simplices of current width strictly less than k are present in the d-dimensional k-Delaunay tree.*

The extension of the construction of the k-Delaunay tree to higher dimensions is straightforward.

The technique of Section 4.1 that deduces the order l Voronoï diagram from the k-Delaunay tree can be extended to higher dimensions in a straightforward manner. As in Section 4.1, we can traverse the graph consisting of the vertices and the edges of the order l Voronoï diagram and compute the labels of each region. From this graph and the labels, it is plain to compute the faces of all dimensions of the diagram.

The algorithm presented in Section 4.2 can also be extended without difficulty, to compute the l nearest neighbours of a given site ($l \leq k$).

5.3 Analysis of the randomized construction

This section presents the following theorem, which generalizes Theorem 3.1 to d dimensions :

Theorem 5.1 *For any set of n sites, if we randomize the sequence of their insertion, the k-Delaunay tree (and thus the order $\leq k$ Voronoï diagrams) of the n sites can be constructed in expected time $O\left(k^{\lceil \frac{d+1}{2} \rceil + 1} n^{\lfloor \frac{d+1}{2} \rfloor}\right)$ using expected storage $O\left(k^{\lceil \frac{d+1}{2} \rceil} n^{\lfloor \frac{d+1}{2} \rfloor}\right)$.*

We only give the main steps achieving the proof of this theorem.

As in Section 3.4, we can define, for $d+2$ sites $X_1, X_2, \ldots, X_{d+2}$, the bicycle $X_1(X_2 \ldots X_{d+1})X_{d+2}$. We also define the width of a simplex and the width of a bicycle.

The following lemmas generalize Lemmas 3.1 to 3.2.

Lemma 5.1 *Let $X_1X_2\dots X_{d+1}$ be a simplex having width j. $X_1X_2\dots X_{d+1}$ will arise as a vertex of some order $\leq k$ Voronoï diagram during the construction with probability :*

$$\begin{cases} \dfrac{k(k+1)\dots(k+d)}{(j+1)(j+2)\dots(j+d+1)} & \text{if } \ j \geq k \\ 1 & \text{if } \ j < k \end{cases}$$

Lemma 5.2 *Let $X_1(X_2\dots X_{d+1})X_{d+2}$ be a bicycle of width j such that $X_2X_3\dots X_{d+2}$ is a son or a stepson of $X_1X_2\dots X_{d+1}$. The probability that such a bicycle appears during the construction is*

$$\frac{(d+1)!}{(j+1)\dots(j+d+2)}$$

As in [10], we have :

Lemma 5.3 *The number of simplices having width at most j is*

$$|T_{\leq j}| = O\left(n^{\lfloor\frac{d+1}{2}\rfloor}(j+1)^{\lceil\frac{d+1}{2}\rceil}\right)$$

Lemma 5.4 *The number of bicycles having width at most j is*

$$|C_{\leq j}| = O\left(n^{\lfloor\frac{d+1}{2}\rfloor}(j+1)^{\lceil\frac{d+1}{2}\rceil+1}\right)$$

Let us now compute the complexity of the k-Delaunay tree.

Proposition 5.1 *The expected space complexity of the k-Delaunay tree is*

$$O\left(k^{\lceil\frac{d+1}{2}\rceil}n^{\lfloor\frac{d+1}{2}\rfloor}\right)$$

Proof : The proof is similar to the one of Proposition 3.1, using Lemmas 5.1 and 5.3, and the fact that, as in 2 dimensions, the total number of links to sons and stepsons is less than the number of nodes (Lemma 2.2), and each node has at most $2(d+1)$ neighbours. □

Proposition 5.2 *The expected cost of Procedure* location *is*

$$O\left(k^{\lceil\frac{d+1}{2}\rceil+1}n^{\lfloor\frac{d+1}{2}\rfloor}\right)$$

Proof : Lemmas 5.2 and 5.4 allow to prove the result, using arguments similar to those used in the proof of Lemma 3.6. □

The cost of Procedure creation is the cost of creating all the vertices of the order $\leq k$ Voronoï diagrams plus the cost of maintaining the neighbourhood relationships, after each insertion of a new site m. The computation of the neighbourhood relationships between the new simplices created by the insertion of a new point m is the same as in the two dimensional case, an edge is simply replaced by a $(d-1)$-face.

So we obtain :

Proposition 5.3 *The expected cost of Procedure* creation *is*

$$O\left(k^{\lceil \frac{d+1}{2} \rceil} n^{\lfloor \frac{d+1}{2} \rfloor} \log k\right)$$

Altogether, Propositions 5.1, 5.2 and 5.3 prove Theorem 5.1.

6 Experimental results

It is to be noted that the algorithm is simple, even if its description and its analysis may look rather intricate ! The core of the algorithm is given in Figures 5 and 6. Moreover, the numerical computations involved are also quite simple : they consist mostly of comparisons of (squared) distances in order to check if a point lies inside or outside a ball. The algorithm has been implemented in the two dimensional case. The program consists of less than 1000 lines of C. It has run on many examples with different kinds of point distributions. Some results and statistics are presented in Figures 9 to 18.

6.1 Influence of randomization

In Figure 9, the points lie on three ellipses, two for the *eyes* and one for the *head*. We tried several permutations of the points for the computation of the 3-Delaunay tree :

1. Points of the head in the order along the ellipse, and then points in the order along each eye.
2. The same as the preceding, except that one point of an eye is inserted first.
3. Points on the eyes in order, and then points on the head in order.
4. A random permutation.
5. Another random permutation.

In Figure 10, the function drawn in dotted line shows the total number of vertices of the order ≤ 3 Voronoï diagrams, versus the number of inserted sites. The functions drawn in bold line show the numbers of nodes in the 3-Delaunay tree, for the different permutations. The functions drawn in thin line show the numbers of nodes visited by the first part of Procedure location (we will call those nodes *1-visited nodes* for short, since they correspond to a traversal of the *(1-)Delaunay tree*), to find a Delaunay triangle in conflict with the new point.

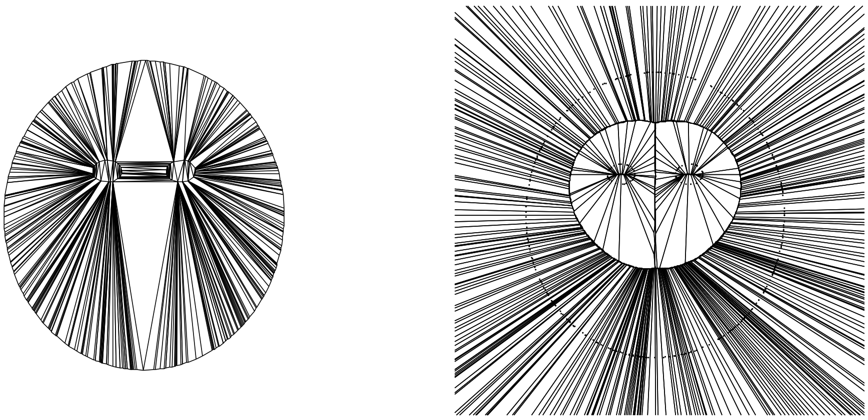


Figure 9: Delaunay triangulation and order 3 Voronoï diagram of a set of 415 points

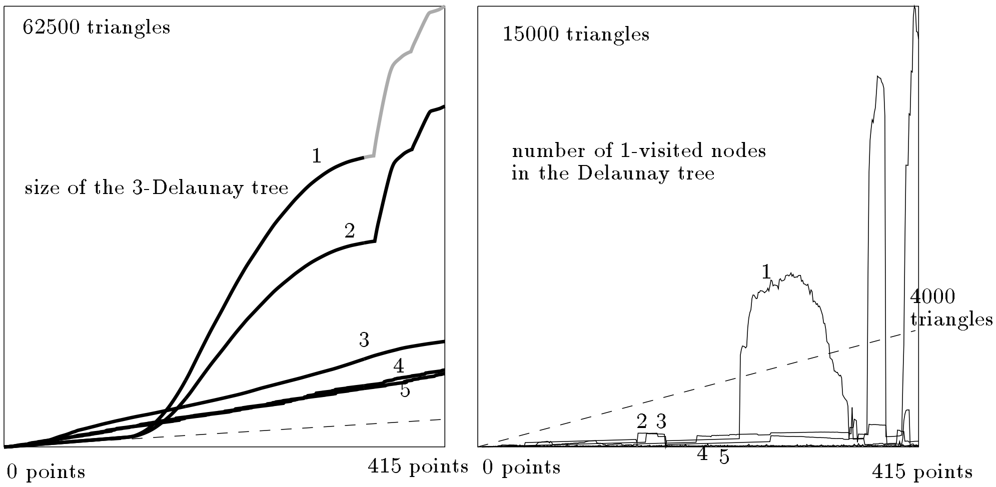


Figure 10: Results for the set of points of Figure 9, $k = 3$, different permutations for the insertion

Permutation	1	2	3	4	5
Size of the 3-Delaunay tree	62522	48334	14958	10903	10395
Size of the ≤ 3 Voronoï	3917	3917	3917	3917	3917
Max nb of 1-visited nodes	14835	1245	1329	181	73
Average nb of 1-visited nodes	2123	245	239	33.5	27
Max nb of created triangles	1044	853	71	210	179
Average nb of created triangles	151	117	36	26.4	25.1

Figure 11: Statistics for the set of points of Figure 9

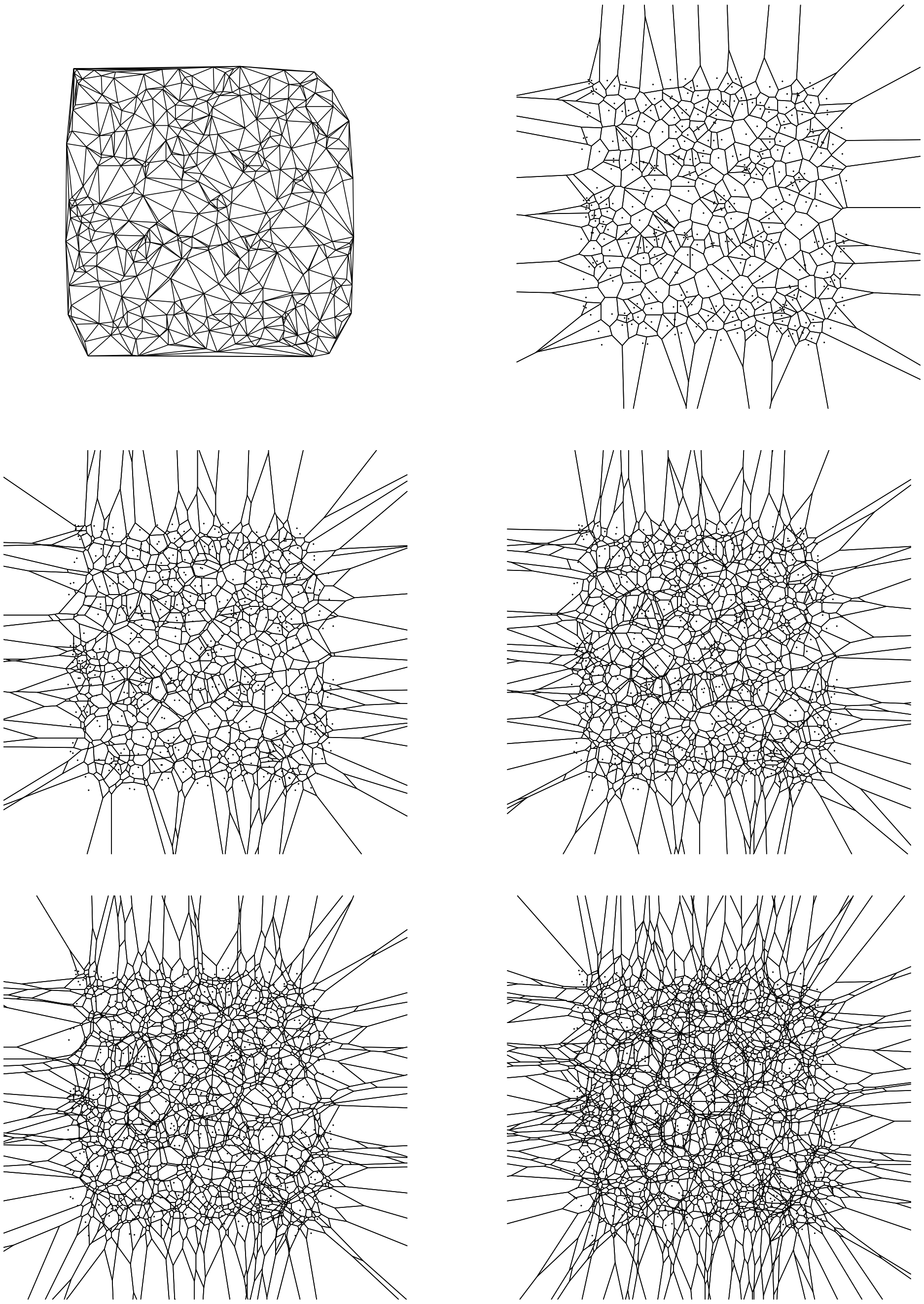


Figure 12: Delaunay triangulation and order 1,2,3,4 and 6 Voronoï diagrams of a set of 400 random points in a square

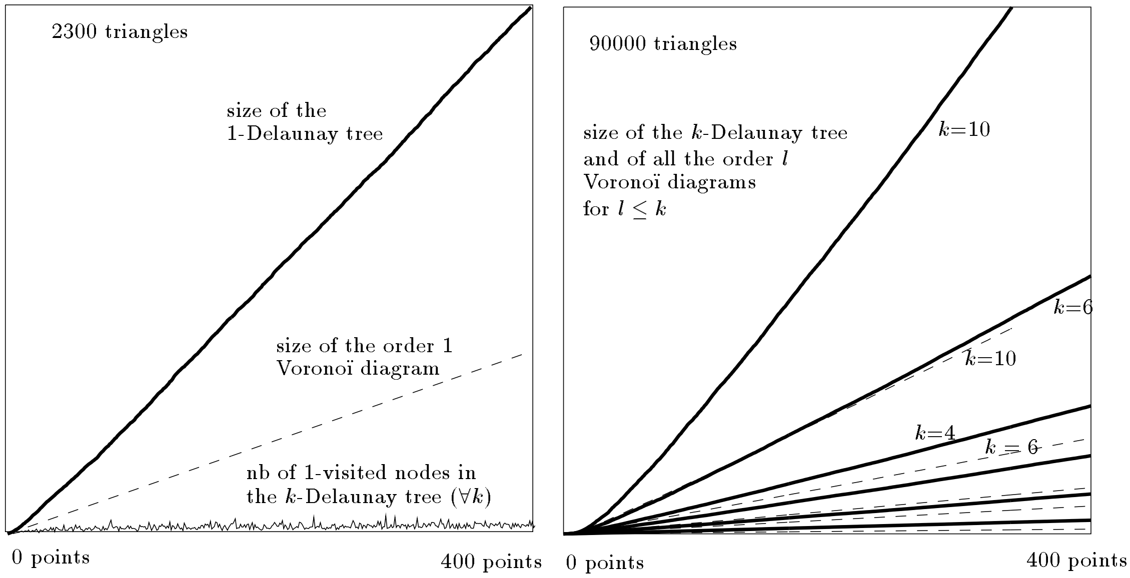


Figure 13: Results for the set of points of Figure 12, for different values of k

k	1	2	3	4	6
Size of the k-Delaunay tree	2307	6748	13246	21694	43740
Size of the $\leq k$ Voronoï	799	2378	4720	7815	16198
Max nb of 1-visited nodes	79	79	79	79	79
Average nb of 1-visited nodes	31	31	31	31	31
Max nb of created triangles	10	29	50	80	150
Average nb of created triangles	5.8	16.9	33.2	54.5	110

Figure 14: Statistics for the set of points of Figure 12

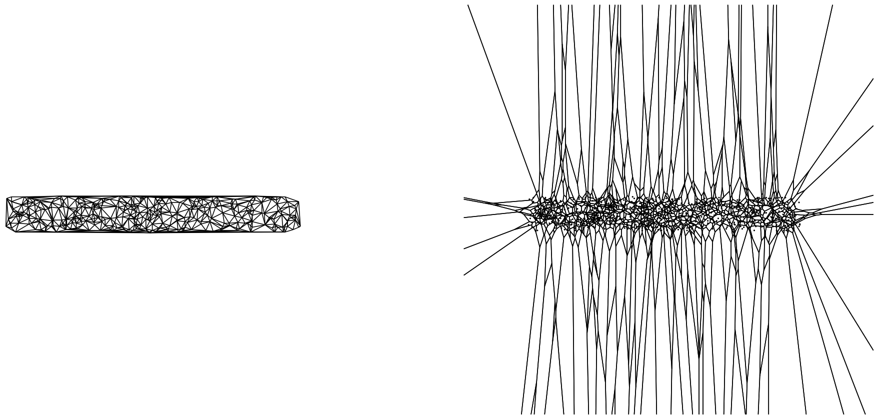


Figure 15: Delaunay triangulation and order 3 Voronoï diagram of a set of 400 random points in a thin rectangle

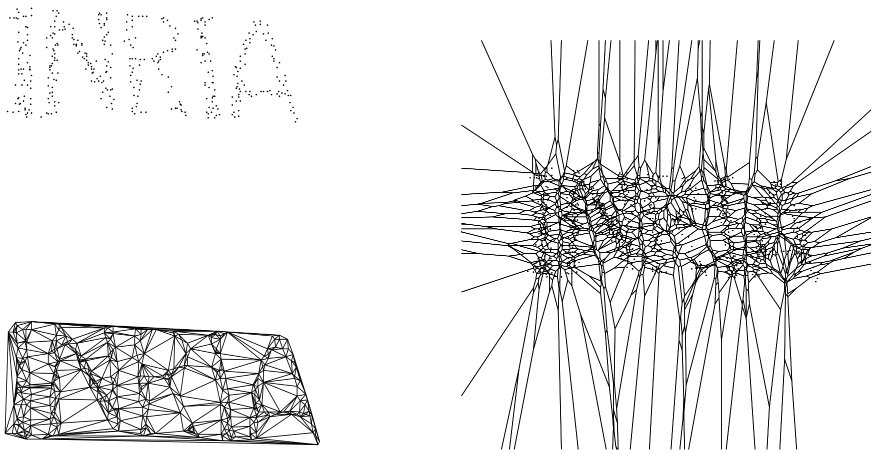


Figure 16: A set of 400 points, the Delaunay triangulation and the order 3 Voronoï diagram

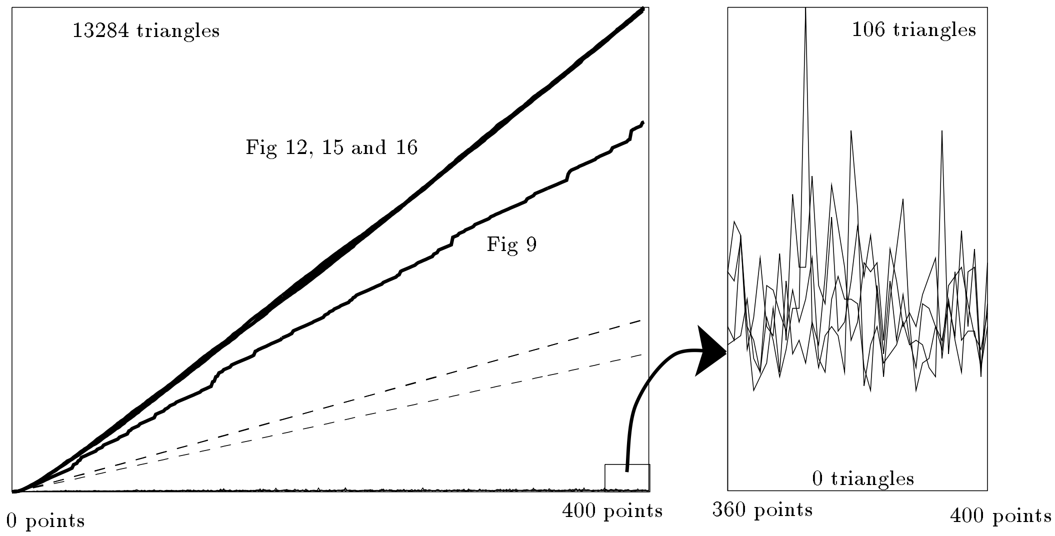


Figure 17: Results for the set of points of Figures 9, 12, 15 and 16 for $k = 3$

Set of points	Fig. 9	Fig. 12	Fig. 15	Fig. 16
Size of the 3-Delaunay tree	10141	13246	13284	13184
Size of the ≤ 3 Voronoï	3776	4720	4718	4718
Max nb of 1-visited nodes	109	79	90	78
Average nb of 1-visited nodes	30	31	33	34
Max nb of created triangles	236	50	67	57
Average nb of created triangles	25.6	33.2	33.1	33.1

Figure 18: Statistics for the sets of points of Figures 9, 12, 15 and 16

Figure 11 gives some statistics about the computation using the different permutations : the size of the 3-Delaunay tree and the sum of the sizes of the order ≤ 3 Voronoï diagrams at the end of the execution, and for the insertion of one point, the maximal and average numbers of 1-visited nodes and created nodes.

This example shows that, if degenerate orders may be really unefficient, the behaviour for random order, or even for the third permutation is satisfying.

6.2 Influence of k

To study the effect of increasing k on the algorithm, we use the set of points depicted in Figure 12.

Figure 13 presents, on the left side, the size of the (1-)Delaunay tree in bold line, the size of the (order 1) Voronoï diagram in dotted line and in thin line the number of nodes that are 1-visited by Procedure location, which does not depend on k. The right side of Figure 13 shows the size of the k-Delaunay tree and the sum of the sizes of the order $\leq k$ Voronoï diagrams for $k =$ 1,2,3,4,6 and 10.

Figure 14 presents statistics similar to those in Figure 11.

6.3 Influence of the point distribution

These last statistics concern the influence of the point distribution. To this aim, we compute the 3-Delaunay tree for the four sets of 400 points described in Figures 9, 12, 15 and 16. Figure 17 shows in this left part the size of the 3-Delaunay tree, and the sum of the sizes of the order ≤ 3 Voronoï diagrams and the number of 1-visited nodes for the four sets. This figure demonstrates that, with a randomization of the input data, the average behaviour of the algorithm does not depend on the distribution of the points. The better result for points of Figure 9 is only due to the smallest size of the output (which is related to the points on the k-hulls). One can see also that the location time is very small in comparison with the sum of the sizes of the ≤ 3 Voronoï diagrams. The right part of Figure 17 shows only the location time for the forty last points. Figure 18 presents statistics as in Figures 11 and 14.

The experimental results show that the algorithm performs well even on degenerate distributions of points.

7 Conclusion

We have shown that the k-Delaunay tree of n points can be constructed in $O(n\log n + k^3 n)$ (*resp.* $O\left(k^{\lceil\frac{d+1}{2}\rceil+1}n^{\lfloor\frac{d+1}{2}\rfloor}\right)$) randomized expected time in the plane (*resp.* in d space). Its randomized expected size is $O(k^2 n)$ (*resp.* $O\left(k^{\lceil\frac{d+1}{2}\rceil}n^{\lfloor\frac{d+1}{2}\rfloor}\right)$). The k-Delaunay tree allows to compute the order $\leq k$ Voronoï diagrams of n points within the same bounds. Any order $l \leq k$ Voronoï diagram can be deduced from the k-Delaunay tree in time proportional to its size, which is $O(ln)$ in two dimensions. Moreover, the k-Delaunay tree can be used to find the l nearest neighbours of a given point.

An important point is that these results hold whatever the point distribution may be.

Our bounds are not as good as those of Mulmuley : the increase by one in the exponent of k is a consequence of the fact that we maintain, at each stage of the incremental insertion, the complete informations relative to all order $\leq k$ Voronoï diagrams, whereas Mulmuley only maintains the vertices of the diagrams, without maintaining their order (which can be deduced at the end of the construction).

The algorithm is simple and, moreover, the numerical computations involved are also quite simple : they consist mostly of comparisons of (squared) distances in order to check if a point lies inside or outside a ball.

Experimental results, for uniform as well as degenerate distributions of points, have provided strong evidence that this algorithm is very effective in practice, for small values of k.

For large values of k, a similar structure, based on the order k furthest neighbours Voronoï diagrams could be derived. It would provide results similar to the ones above to construct all order $\geq n-k$ Voronoï diagrams and to find l furthest neighbours for $l \leq k$.

The Delaunay tree can also be generalized to many other problems, such as the construction of Voronoï diagrams of segments, and the computation of arrangements. Results are reported in a forthcoming paper [17].

Acknowledgements

It is a pleasure to thank the two referees whose comments improved significantly the quality of the paper. We would also like to thank Jean-Pierre Merlet for supplying us with his interactive drawing preparation system JPdraw.

References

[1] J-D. Boissonnat and M. Teillaud. A hierarchical representation of objects: The Delaunay Tree. In *Second ACM Symposium on Computational Geometry in Yorktown Heights*, June 1986.

[2] J-D. Boissonnat and M. Teillaud. On the randomized construction of the Delaunay tree. *Theoretical Computer Science.* To appear. Available as Technical Report INRIA 1140.

[3] M.I. Shamos and D. Hoey. Closest-point problems. In *IEEE Symposium on Foundations of Computer Science*, pages 151–162, October 1975.

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

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

[6] B. Chazelle and H. Edelsbrunner. An improved algorithm for constructing k^{th}-order Voronoi diagrams. In *First ACM Symposium on Computational Geometry in Baltimore*, pages 228–234, June 1985.

[7] K.L. Clarkson. New applications of random sampling in computational geometry. *Discrete and Computational Geometry*, 2:195–222, 1987.

[8] L.J. Guibas, D.E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7(4):381–413, 1992.

[9] K. Mehlhorn, S. Meiser, and C. Ó'Dúnlaing. On the construction of abstract Voronoi diagrams. *Discrete and Computational Geometry*, 6:211–224, 1991.

[10] K.L. Clarkson and P.W. Shor. Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry*, 4(5), 1989.

[11] K. Mulmuley. On obstruction in relation to a fixed viewpoint. In *IEEE Symposium on Foundations of Computer Science*, pages 592–597, 1989.

[12] R.A. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. *Discrete and Computational Geometry*, 6:343–367, 1991.

[13] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, 15:341–363, 1986.

[14] K. Mulmuley. On levels in arrangements and Voronoi diagrams. *Discrete and Computational Geometry*, 6:307–338, 1991.

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

[16] P.J. Green and R. Sibson. Computing Dirichlet tesselations in the plane. *The Computer Journal*, 21, 1978.

[17] 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*, 8:51–71, 1992.