ELSEVIER

Computational Geometry 17 (2000) 69–96

Computational Geometry
Theory and Applications

www. elsevier.nl/locate/comgeo

# A probabilistic result on multi-dimensional Delaunay triangulations, and its application to the 2D case

C. Lemaire [a,1], J.-M. Moreau [b,*]

[a] *CEA, Bruyères, France*
[b] *LISSE/ENSM.SE, St-Étienne, France*

Communicated by K. Mehlhorn; received 8 February 2000; revised 13 July 2000; accepted 24 July 2000

**Abstract**

This paper exploits the notion of "unfinished site", introduced by Katajainen and Koppinen (1998) in the analysis of a two-dimensional Delaunay triangulation algorithm, based on a regular grid. We generalize the notion and its properties to any dimension $k \geqslant 2$: in the case of uniform distributions, the expected number of unfinished sites in a $k$-rectangle is $O(N^{1-1/k})$. This implies, under some specific assumptions, the linearity of a class of divide-and-conquer schemes based on balanced k-d trees.

This general result is then applied to the analysis of a new algorithm for constructing Delaunay triangulations in the plane. According to Su and Drysdale (1995, 1997), the best known algorithms for this problem run in linear expected time, thanks in particular to the use of bucketing techniques to partition the domain. In our algorithm, the partitioning is based on a 2-d tree instead, the construction of which takes $\Theta(N \log N)$ time, and we show that the rest of the algorithm runs in linear expected time. This "preprocessing" allows the algorithm to adapt efficiently to irregular distributions, as the domain is partitioned using point coordinates, as opposed to a fixed, regular basis (buckets or grid). We checked that even for the largest data sets that could fit in internal memory (over 10 million points), constructing the 2-d tree takes noticeably less CPU time than triangulating the data. With this in mind, our algorithm is only slightly slower than the reputedly best algorithms on *uniform* distributions, and is even the most efficient for data sets of up to several millions of points distributed in *clusters*. © 2000 Elsevier Science B.V. All rights reserved.

*Keywords:* Delaunay triangulation; k-d tree; Worst-case complexity; Average-case complexity; Multi-dimensional divide-and-conquer

* Corresponding author.
*E-mail addresses:* lemairec@bruyeres.cea.fr (C. Lemaire), moreau@emse.fr, jmmoreau@ligim.univ-lyon1.fr (J.-M. Moreau).

[1] Research conducted while the author was employed at SETRA.


PII: S0925-7721(00)00017-1

## 1. Introduction

### 1.1. General outline

The paper is divided into four distinct parts (notwithstanding a rather technical appendix at the end):
(1) Introduction.
(2) Study of a general probabilistic result on multi-dimensional Delaunay triangulations.
(3) Application to the planar case.
(4) Perspectives and conclusion.

### 1.2. A short survey of the past

The Voronoi diagram of a set of $N$ distinct points in the plane is a well-studied problem [3], for which Shamos and Hoey published the first optimal *divide-and-conquer* solution [31]. Since the Delaunay triangulation of the same set is $O(N)$-reducible from the former structure by duality, the authors thus proved that the (indirect) construction of the latter could be performed in $\Theta(N \log N)$, which is *worst-case* optimal under the *real-RAM* model of computation. The first direct worst-case optimal 2D-Delaunay construction divide-and-conquer algorithm was published by Lee and Schachter [23], to be later refined by Guibas and Stolfi [20].

In 1984, Ohya et al. [21] proved the *expected* running time of these divide-and-conquer algorithms to be $\Omega(N \log N)$ when the sites are uniformly distributed in the unit square. However, several authors studied the possibility to break that bound: in 1984, Ohya et al. themselves [21] and Maus [27] published independent $[O(N), O(N^2)]$ expected/worst-case methods based on the so-called *bucketing technique* (partition of the domain into $N$ cells). In 1987, Dwyer published an $[O(N \log\log N), O(N \log N)]$ expected/worst-case variant [13] of Lee and Schachter's algorithm, in which the domain is partitioned into $N/\log N$ cells, that are triangulated, the triangulations in each row are then combined, and the resulting triangulations are combined together.

In their 1988 paper [22], Katajainen and Koppinen modified Dwyer's algorithm to achieve linear expected time (partition into $N$ cells, merge of cells in a quadtree-like order). Their average-case analysis exploited the notion of *unfinished sites* in a rectangular domain. Finally, Dwyer published the first $k$-dimensional method with expected linear behaviour in 1991 [14], an incremental algorithm also based on buckets.

### 1.3. A finer outline of the paper

Section 2 goes one step further in this progression: after stating some definitions and general assumptions on the distribution of sites (Section 2.1), we generalize the notion of unfinished site to any dimension (Section 2.2), and we establish probabilistic results in the case of quasi-uniform distributions: probability for a site to be unfinished, expected number of unfinished sites in a hyperrectangle (Sections 2.3–2.5). The upper bounds we provide are valid in any dimension, and the constants we obtain for the 2D case are tighter than the original ones established by Katajainen and Koppinen.

Next, we present a divide-and-conquer scheme based on a (balanced) k-d tree [5] (Section 2.6), and show that, in the case of a quasi-uniform distribution in a hypercube, the expected running time of the whole multi-dimensional merge phase is linear, if merging two subsets is assumed to take time

proportional to the number of unfinished sites they contain. These results may be used to analyze the running times of some classes of $k$-dimensional divide-and-conquer algorithms, for which the notion of unfinished site has a meaning, and are hence more or less directly related to the Delaunay triangulation.

In Section 3, this general result is used to analyze a new algorithm to construct the Delaunay triangulation in the plane [25,26] (Section 3.3), that is shown, in the first author's Ph.D. Dissertation [24], to be among the most efficient. Section 3.4 then gives experimental results on this algorithm, and a comparison with the best known algorithms, based on Su and Drysdale's study [35,36].

The paper closes on potential extensions of the results presented in this paper, and other related problems (Section 4).

## 2. A probabilistic result

### 2.1. Quasi-uniform assumption

Throughout this paper, we shall consider sets of $N$ points (also called *sites*) in a Euclidean space of dimension $k \geqslant 2$, and we shall assume the sites to be *quasi-uniformly* distributed in a unit cube $\mathcal{U}_k$. This implies the existence of two strictly positive real constants $c_1 \leqslant c_2$, and of a probability density $f$ such that

$$\begin{cases} \forall (x_1, x_2, \ldots, x_k) \in \mathcal{U}_k, & c_1 \leqslant f(x_1, x_2, \ldots, x_k) \leqslant c_2, \\ \forall (x_1, x_2, \ldots, x_k) \notin \mathcal{U}_k, & f(x_1, x_2, \ldots, x_k) = 0. \end{cases}$$

Accordingly, the probability for one given site to lie in domain $\mathcal{D}$ is $\int_{\mathcal{D}} f$. Uniform distribution may be seen as a particular case of the more general definition of quasi-uniform distribution.

### 2.2. Unfinished sites

The notion of *unfinished site* was introduced by Katajainen and Koppinen [22] for the analysis of an algorithm to construct Delaunay triangulations in the plane, that partitions the domain according to a regular grid. Consider, for instance, the planar Delaunay triangulation, $DT(S_1)$, of the point set $S_1$, subset of $S$ (refer to Fig. 1). Several sites in $S_1$ have the same adjacency list in the whole Delaunay triangulation, $DT(S)$, as in sub-triangulation $DT(S_1)$: such sites are said to be *finished* in $DT(S_1)$ with respect to $DT(S)$. By opposition, the sites (white circles in the same figure) that receive or lose edges are said to be *unfinished* (in $DT(S_1)$ with respect to $DT(S)$). Intuitively, unfinished sites cannot lie very far from the section of convex hull boundary of $DT(S_1)$ facing the complement of $S_1$ in $S$. In the sequel we shall give an upper bound on the probability for a site in a rectangular domain to be unfinished, in the case of a quasi-uniform distribution. This upper bound increases as the site gets closer to the boundary of the rectangle that contains it. We shall also give an upper bound on the expected number of unfinished sites inside a rectangular domain.

We now present a generalization of these notions to any dimension, and their mathematical formulation. $DT(S)$ will denote the $k$-Delaunay triangulation of a set of sites $S$.

**Definition 2.1.** Let $T(S_1)$ and $T(S_2)$ be the triangulations of sets $S_1$ and $S_2$. We shall write $T(S_1) <_\Delta T(S_2)$ whenever $S_1 \subseteq S_2$ and $T(S_1)$ contains each edge in $T(S_2)$ the endpoints of which belong to $S_1$.

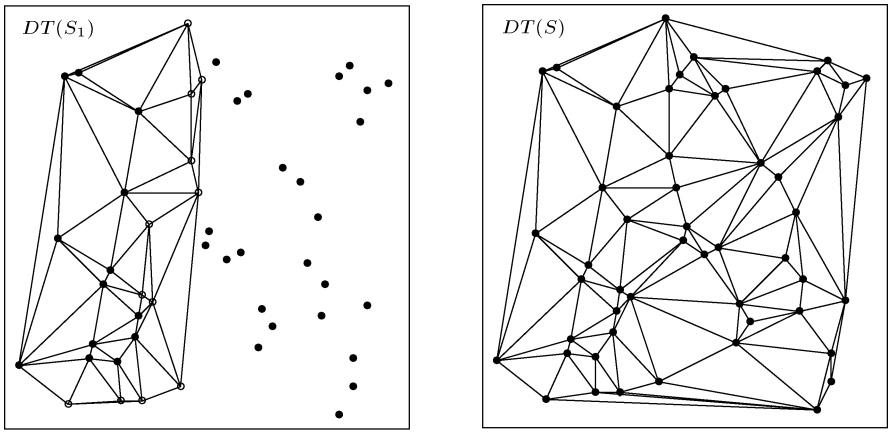


Fig. 1. Unfinished sites in a sub-triangulation.

**Definition 2.2.** Let $T(S_1) <_\Delta T(S_2)$ be two triangulations and $s \in S_1$. We say that site $s$ is *finished* in $T(S_1)$ with respect to $T(S_2)$ if the set of edges adjacent to $s$ in $T(S_1)$ and $T(S_2)$ coincide; otherwise, $s$ is said to be *unfinished* (in $T(S_1)$ with respect to $T(S_2)$).

The first definition induces a partial order in the set of triangulations. Notice that $T(S_1) <_\Delta T(S_2)$ does not imply that all edges in $T(S_1)$ belong to $T(S_2)$. Also note that $DT(S_1) <_\Delta DT(S_2)$ is equivalent to $S_1 \subseteq S_2$ if $S_1$ has a unique Delaunay triangulation.

**Proposition 2.3.** *Let $T(S_1) <_\Delta T(S_2)$ be two triangulations. A site $s \in S_1$ is finished in $T(S_1)$ with respect to $T(S_2)$ if and only if $S_1$ contains the endpoints of all edges adjacent to $s$ in $T(S_2)$.*

**Proof.** Let $(s, p_1), \ldots, (s, p_m)$ be the edges adjacent to $s$ in $T(S_2)$. If $s$ is finished, then $p_i$'s endpoints are in $S_1$ by definition. Conversely, suppose that $p_i$'s endpoints are in $S_1$ but that $s$ is not finished. Then, $(s, p_1), \ldots, (s, p_m)$ are in $T(S_1)$ because $T(S_1) <_\Delta T(S_2)$, but $T(S_1)$ contains another edge $(s, r)$, at least. Since the domain limited by $T(S_2)$ is convex, it contains edge $(s, r)$ completely. Hence, $(s, r)\backslash\{s\}$ crosses either:

- one of the open $k$-triangles from $T(S_2)$ that share $s$ as vertex; since no $k$-triangle from $T(S_2)$ contains any site in its interior, none contains $r$, and hence, $(s, r)$ crosses one $k$-face opposite $s$; however, this is impossible, since this $k$-face also belongs to $T(S_1)$, which is a triangulation,
- or one $k$-face containing $s$, which is not possible since this $k$-face also belongs to $T(S_1)$, which is a triangulation. □

**Corollary 2.4.** *Let $T(S_1) <_\Delta T(S_2) <_\Delta T(S_3)$ be three triangulations. A site $s \in S_1$ is finished in $T(S_1)$ with respect to $T(S_3)$ if and only if it is both finished in $T(S_1)$ with respect to $T(S_2)$, and in $T(S_2)$ with respect to $T(S_3)$.*

**Proof.** That the latter implies the former is a consequence of Definition 2.2. Conversely, suppose $s$ is finished in $T(S_1)$ with respect to $T(S_3)$. If $(s,q)$ is one edge in $T(S_3)$, $q \in S_1$; since $q \in S_2$, and, using Proposition 2.3, we may conclude that $s$ is finished in $T(S_2)$ with respect to $T(S_3)$. Finally, because of Definition 2.2, $s$ is also finished in $T(S_1)$ with respect to $T(S_2)$. □

### *2.3. k-partition around an unfinished site*

In order to compute probabilities related to a given site, it is necessary to build *a fixed* geometric "paving" around this point. The volume of each paving block will then represent a probability, up to a multiplicative constant.

**Lemma 2.5.** *If $s$ is an unfinished site in $k$-rectangle $\mathcal{R}_k$, at distance $t$ from the boundary of $\mathcal{R}_k$, then there is a $k$-ball $\mathcal{B}_2$ with radius $t/2$, center $p$ at distance $t/2$ from $s$, and no sites in its interior.*

**Proof.** Referring to Fig. 2, if $s$ is an unfinished site in $k$-rectangle $\mathcal{R}_k$, there is a site $q$ belonging to $\mathcal{U}_k \backslash \mathcal{R}_k$ such that $(s,q)$ is a Delaunay edge in $\mathcal{U}_k$. Since $q$ lies outside $k$-rectangle $\mathcal{R}_k$, the length of $(s,q)$ is greater than $t$. And since $(s,q)$ is a Delaunay edge, we may find a $k$-ball $\mathcal{B}_3$ with no sites in its interior, and $s$ and $q$ on its boundary. The diameter of this $k$-ball is thus greater than $t$. Let $c$ be the center of $\mathcal{B}_3$, and $\mathcal{B}_2$ be the $k$-ball with center $p$ lying on segment $[sc]$ at distance $t/2$ from $s$ (cf. Figs. 2 and 3). $\mathcal{B}_2$ is contained in $\mathcal{B}_3$, hence $\mathcal{B}_3$ has no sites in its interior. □

**Corollary 2.6** (Refer to Fig. 3). *In the setting of Lemma 2.5, let $\mathcal{B}_1(s,r_1)$ be the ball centered in $s$ with radius $r_1 < t$, and $\mathcal{C}(sp,\theta)$ be the cone with apex $s$, symmetry axis $sp$, and angle $\theta = \arccos(r_1/t)$. Then, $\mathcal{C}_{\mathcal{B}_1} = \mathcal{C} \cap \mathcal{B}_1$ is empty of sites.*

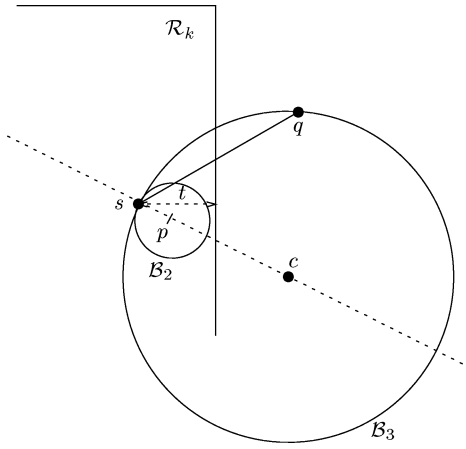


Fig. 2. Ball $\mathcal{B}_2$ has no sites in its interior.

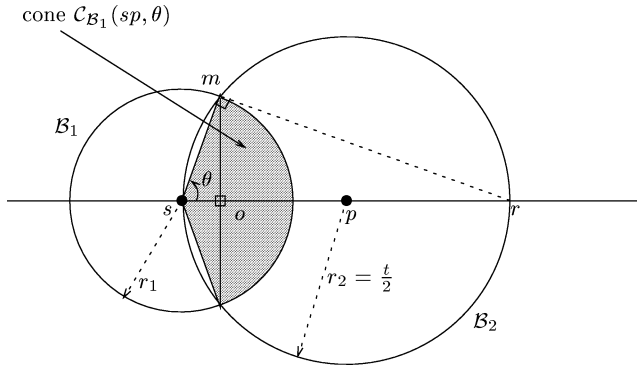


Fig. 3. Cone $\mathcal{C}_{\mathcal{B}_1}(sp,\theta)$ has no sites in its interior.

**Proof.** Let $m$ be any point on the intersection of the boundaries of balls $\mathcal{B}_1$ and $\mathcal{B}_2$. Let $o$ be the orthogonal projection of $m$ on segment $[sp]$, $r$ diametrically opposite $s$ on ball $\mathcal{B}_2$, and let $\theta$ be the angle formed by segments $[so]$ and $[sm]$.

$$\cos\theta = \frac{sm}{sr} = \frac{r_1}{t} \quad \Rightarrow \quad \theta = \arccos\frac{r_1}{t}.$$

The cone portion $\mathcal{C}_{\mathcal{B}_1}(sp,\theta)$ is included in $\mathcal{B}_2$, hence $\mathcal{C}_{\mathcal{B}_1}(sp,\theta)$ is empty of sites. □

*Notation convention*. We shall write $\langle p_1, p_2, p_3\rangle$ to denote the angle made by the three points $p_1$, $p_2$ and $p_3$, with $p_2$ at the apex.

**Lemma 2.7.** *Consider a hypercube inscribed in a ball $\mathcal{B}_1(s,r_1)$ with center $s$ and radius $r_1$. Each face $\mathcal{F}$ of the hypercube is partitioned into "cells"* (*i.e., $(k-1)$-cubes*) *by all the hyperplanes parallel to* other *faces of the hypercube and containing the symmetry center of face $\mathcal{F}$. Let us call pyramid $\mathcal{P}$ the intersection with ball $\mathcal{B}_1$ of the cone issued from $s$, and having one of the above "cells" as section. We have*

(1) *ball $\mathcal{B}_1$ is partitioned into $(k2^k)$ pyramids that all have the same volume,*

(2) $\forall M_1, M_2 \in \mathcal{P}\backslash\{s\}$,

$$\cos\langle M_1, s, M_2\rangle \geqslant \left(\left\lceil\frac{k+1}{2}\right\rceil\left\lfloor\frac{k+1}{2}\right\rfloor\right)^{-1/2}.$$

**Proof.** (1) In $k$-space, a hypercube is a regular polyhedron with $(2k)$ faces, that may be inscribed in a $k$-ball, and this in any dimension [6,7]. Each face $\mathcal{F}$ undergoes $(k-1)$ divisions and is thus decomposed into $2^{k-1}$ "cells" (i.e., $(k-1)$-cubes). Hence, there are $2k \times 2^{k-1} = k2^k$ such cells, on each of which one pyramid may be constructed. Thus, ball $\mathcal{B}_1$ is partitioned into $(k2^k)$ pyramids, that, for symmetry reasons, all have the same volume:

$$\mathcal{V}(\mathcal{P}) = \frac{\mathcal{V}(\mathcal{B}_1)}{k2^k} = \frac{\pi^{k/2}r_1^k}{k2^k\Gamma(k/2+1)}. \tag{1}$$

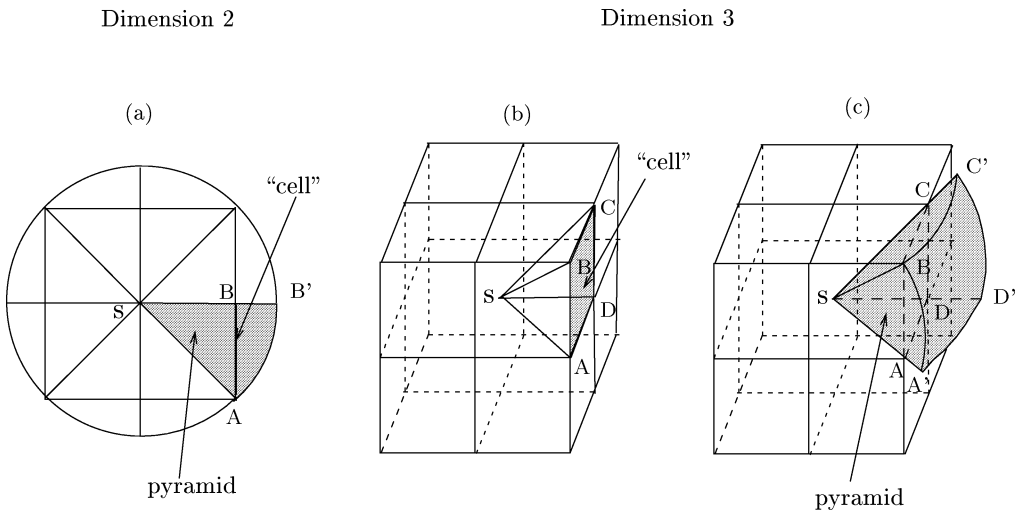


Fig. 4. Decomposition of a ball into pyramids.

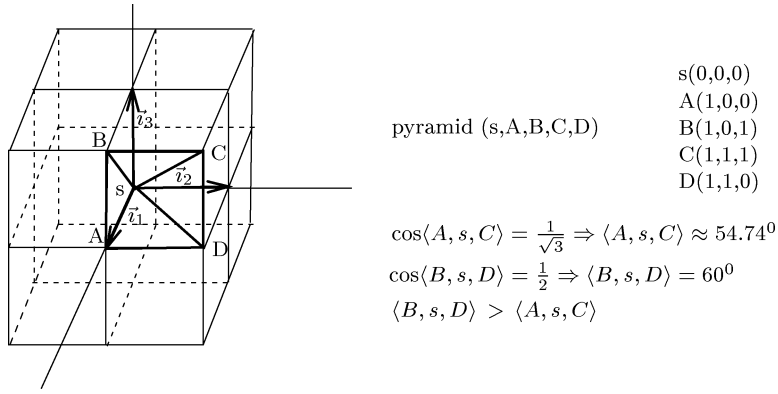


Fig. 5. Maximum angle at the apex of a pyramid.

In dimension 2 (cf. Fig. 4(a)), each side of the square is divided into two equal segments, which yields 8 cells, each supporting an angular-sector pyramid. In dimension 3 (cf. Fig. 4(b)), each face of the cube is divided into 4 equal squares, which yields 24 cells, each supporting one pyramid (cf. Fig. 4(c)), etc.

(2) Let $(s, \vec{\imath}_1, \ldots, \vec{\imath}_k)$ be an orthogonal system of coordinates, with axes perpendicular to the faces of the hypercube and norms equal to half the side of the hypercube. Without loss of generality, consider the "cell" contained in the hyperplane $x_1 = 1$ and such that the coordinates of all of its vertices are nonnegative. Let $\mathcal{P}$ be the pyramid associated to this cell (refer to Fig. 5 for dimension 3).

Let $M_1(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_k)$ and $M_2(\beta_1, \beta_2, \beta_3, \ldots, \beta_k)$ be any two points in $\mathcal{P}\backslash\{s\}$. We wish to compute the maximum of angle $\langle M_1, s, M_2\rangle$. To that effect, it suffices to let $M_1$ and $M_2$ span all vertices of the "cell", section of the pyramid. This implies that $\alpha_1 = \beta_1 = 1$, while all other $\alpha_i$'s and $\beta_i$'s are either 0 or 1, $i \in [2, k]$.

Whatever the choice for $M_1$ and $M_2$, there are, say, $u \in [1,k]$ coordinates that are 1 in both (at least 1, since $\alpha_1 = \beta_1 = 1$), $v \in [0, k-u]$ that are 1 in $M_1$ and 0 in $M_2$, and finally $w \in [0, k-u-v]$ that are 0 in $M_1$ and 1 in $M_2$ (leaving out those that are 0 in both ...).

Hence, the maximum of $\langle M_1, s, M_2\rangle$ corresponds to the minimum of

$$\cos\langle M_1, s, M_2\rangle = \frac{\sum_{i=1}^{k}\alpha_i\beta_i}{\sqrt{\sum_{i=1}^{k}\alpha_i^2}\sqrt{\sum_{i=1}^{k}\beta_i^2}} = \frac{u}{\sqrt{u+v}\sqrt{u+w}}. \tag{2}$$

The numerator, $u$, may take integral values in the range $[1,k]$. Now, suppose $u \geqslant 1$ is fixed. Since $w \in [0, k-u-v]$, $w \rightsquigarrow u+w$ is a strictly increasing function with maximum value $k-v$, and the maximum of $v \rightsquigarrow \sqrt{u+v}\sqrt{k-v}$ is reached for $v = \lfloor (k-u)/2 \rfloor$.

Finally, since

$$u \rightsquigarrow \frac{u}{\sqrt{u+\lfloor (k-u)/2\rfloor}\sqrt{k-\lfloor (k-u)/2\rfloor}}$$

is an increasing function on $[1,k]$ that reaches its minimum for $u=1$, the minimum of (2) is

$$\frac{1}{\sqrt{1+\lfloor (k-1)/2\rfloor}\sqrt{k-\lfloor (k-1)/2\rfloor}}.$$

After unifying cases $k$ odd and even, we find

$$\forall M_1, M_2 \in \mathcal{P}\backslash\{s\}, \quad \cos\langle M_1, s, M_2\rangle \geqslant \frac{1}{\lfloor (k+1)/2\rfloor^{1/2}\lceil (k+1)/2\rceil^{1/2}}. \tag{3}$$

This result may be checked in the plane and in 3-space, in Figs. 4 and 5, respectively. □

*Notation convention*. In order to simplify the expression of subsequent results, let $\zeta_k$ represent expression $(\lfloor (k+1)/2\rfloor \lceil (k+1)/2\rceil)^{1/2}$.

**Corollary 2.8.** *If $s$ is an unfinished site inside $k$-rectangle $\mathcal{R}_k$, and at distance $t$ from its boundary, then we may find a fixed $k$-partition of the neighbourhood of $s$ with $(k2^k)$ $k$-cells of equal volume,*

$$\frac{\pi^{k/2}t^k}{k2^k\zeta_k^k\Gamma(k/2+1)},$$

*and such that at least one of these partitioning $k$-cells is site-free.*

**Proof.** In the setting of Corollary 2.6, let us fix the radius of ball $\mathcal{B}_1$ to $r_1 = t/\zeta_k$. $\mathcal{B}_1$ is partitioned into pyramids as explained in Lemma 2.7. Let $q$ be the intersection between segment $[sp]$ and the boundary of ball $\mathcal{B}_1$. There exists one pyramid $\mathcal{P}_j$ such that $[sq] \subset \mathcal{P}_j$. Then, for all $M$ in $\mathcal{P}_j$,

$$M \in \mathcal{P}_j \quad \Rightarrow \quad \cos\langle q, s, M\rangle \geqslant \frac{1}{\zeta_k} = \frac{r_1}{t} \quad \Rightarrow \quad M \in \mathcal{C}_{\mathcal{B}_1}\left(sp, \arccos\frac{r_1}{t}\right).$$

This implies that

$$\mathcal{P}_j \subset \mathcal{C}_{\mathcal{B}_1}\left(sp, \arccos\frac{r_1}{t}\right).$$

As the portion of cone $\mathcal{C}_{\mathcal{B}_1}$ is site free, so is the paving block $\mathcal{P}_j$, with volume

$$\frac{\pi^{k/2}t^k}{k2^k\zeta_k^k\Gamma(k/2+1)}. \qquad \square$$

*2.4. Probability for a site to be unfinished*

The paving we have constructed around site $s$ allows us to derive an upper bound on the probability for $s$ to be unfinished.

**Lemma 2.9.** *Assume the probability density $f$ to be quasi-uniform with bounds $c_1$ and $c_2$, and consider rectangle $\mathcal{R}_k \subseteq \mathcal{U}_k$ in $DT(S\cap\mathcal{R}_k) <_\Delta DT(S)$. Let $s \in S\cap\mathcal{R}_k$ be a site at minimal distance $t$ from the boundary of $\mathcal{R}_k$, and let $C_1$ be the condition that $s$ is unfinished in $DT(S\cap\mathcal{R}_k)$ with respect to $DT(S)$, then*

$$\Pr\{\mathcal{C}_1\} \leqslant k2^k\left[1-\frac{c_1\pi^{k/2}t^k}{k2^k\zeta_k^k\Gamma(k/2+1)}\right]^{N-1}.$$

**Proof.** Consider ball $\mathcal{B}_1$, with center $s$ and radius $t/\zeta_k$, paved by pyramids $\mathcal{P}_j$, $j\in[1,k\times 2^k]$. Using Corollary 2.8, condition $\mathcal{C}_1$ – that $s$ is unfinished in $DT(S\cap\mathcal{U}_k)$ with respect to $DT(S)$ – implies condition $\mathcal{C}_2$ – that at least one of the open partitioning $k$-cells $\mathcal{P}_j$ is void of sites. Hence,

$$\Pr\{\mathcal{C}_1\} \leqslant \Pr\{\mathcal{C}_2\} \leqslant \sum_{j=1}^{k2^k}\Pr\{S\cap\mathcal{P}_j=\emptyset\} = \sum_{j=1}^{k2^k}\left(1-\int_{\mathcal{P}_j} f\right)^{N-1} \leqslant k2^k\left[1-\frac{c_1\pi^{k/2}t^k}{k2^k\zeta_k^k\Gamma(k/2+1)}\right]^{N-1},$$

using property $f(x_1,x_2,\ldots,x_k)\geqslant c_1$. $\square$

*2.5. Expected number of unfinished sites*

We are now ready for the main result in this section: a bound on the expected number of unfinished sites in a hyperrectangle.

**Theorem 2.10.** *In the setting of Lemma 2.9, if $\mathcal{S}_k$ is the surface of $\mathcal{R}_k$, and $E(\mathcal{R}_k)$ is the expected number of unfinished sites in $\mathcal{R}_k$, then*

$$E(\mathcal{R}_k) \leqslant \frac{N}{\sqrt[k]{N}}\,\frac{c_2}{\sqrt[k]{c_1}}\,\mathcal{S}_k\frac{k2^{k+1}\zeta_k(k\Gamma(k/2+1))^{1/k}}{\sqrt{\pi}}. \tag{4}$$

**Proof.** Let $a_1,a_2,\ldots,a_k$ be the lengths of the sides of $k$-rectangle $\mathcal{R}_k$. Let $a_l$ be the length of the smallest side of $\mathcal{R}_k$. Consider the elementary volume $\mathcal{V}_t$ comprising the points of $\mathcal{R}_k$ at a distance between $t$ and $t+\mathrm{d}t$ from the boundary of $\mathcal{R}_k$ (with $0\leqslant t\leqslant a_l/2$) (Fig. 6). Let us call $\mathcal{S}_t$ the exterior surface of $\mathcal{V}_t$,

$$\mathcal{S}_t = 2\sum_{j=1}^{k}\left(\prod_{i\neq j}(a_i-2t)\right) \leqslant \mathcal{S}_k = 2\sum_{j=1}^{k}\prod_{i\neq j}a_i,$$

which implies

$$\mathcal{V}_t \leqslant \mathcal{S}_t\,\mathrm{d}t \leqslant \mathcal{S}_k\,\mathrm{d}t.$$

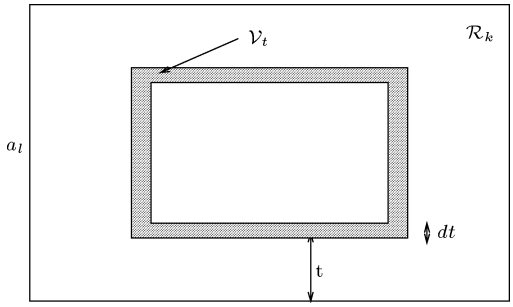


Fig. 6. Expected number of unfinished sites in a $k$-rectangle.

The probability for a given site to belong to $\mathcal{V}_t$ is

$$\int_{\mathcal{V}_t} f \leqslant c_2 \mathcal{V}_t \leqslant c_2 \mathcal{S}_k \, \mathrm{d}t.$$

Using Lemma 2.9, we have the following estimate:

$$\begin{aligned} E(\mathcal{R}_k) &\leqslant N \int_0^{a_l/2} k 2^k \left[1 - \frac{c_1 \pi^{k/2} t^k}{k 2^k \zeta_k^k \Gamma(k/2+1)}\right]^{N-1} c_2 \mathcal{S}_k \, \mathrm{d}t \\ &= N \frac{c_2}{\sqrt[k]{c_1}} \mathcal{S}_k \frac{k 2^{k+1} \zeta_k (k\Gamma(k/2+1))^{1/k}}{\sqrt{\pi}} \int_0^{(a_l \sqrt{\pi/4} \sqrt[k]{c_1})/(\zeta_k (k\Gamma(k/2+1))^{1/k})} \left(1 - x^k\right)^{N-1} \mathrm{d}x, \end{aligned}$$

with the change of variable

$$x = t \left[\frac{c_1 \pi^{k/2}}{k 2^k \zeta_k^k \Gamma(k/2+1)}\right]^{1/k}.$$

Since $a_l \leqslant 1$, $c_1 \leqslant 1$ and $k \geqslant 2$, we may write

$$E(\mathcal{R}_k) \leqslant N \frac{c_2}{\sqrt[k]{c_1}} \mathcal{S}_k \frac{k 2^{k+1} \zeta_k [k\Gamma(k/2+1)]^{1/k}}{\sqrt{\pi}} \int_0^1 \left(1 - x^k\right)^{N-1} \mathrm{d}x.$$

The integral on the right-hand side may be bounded above by $1/\sqrt[k]{N}$ (a rather technical result proven in Appendix A). Hence,

$$E(\mathcal{R}_k) \leqslant \frac{N}{\sqrt[k]{N}} \frac{c_2}{\sqrt[k]{c_1}} \mathcal{S}_k \frac{1}{\sqrt{\pi}} k 2^{k+1} \zeta_k \left[k\Gamma\left(\frac{k}{2}+1\right)\right]^{1/k} = \mathrm{O}\left(\frac{N}{\sqrt[k]{N}} \frac{c_2}{\sqrt[k]{c_1}} \mathcal{S}_k\right) = \mathrm{O}(N^{1-1/k}). \qquad \square$$

Note that the constant factor in this bound is tighter than the one obtained by Katajainen and Koppinen in the plane [22].

### *2.6. Average-case analysis of a class of k-dimensional divide-and-conquer algorithms using the notion of unfinished sites*

In this subsection, we show how the previous result may be exploited to analyze certain classes of multi-dimensional *divide-and-conquer* algorithms. In what follows, we shall use the generic term "conquer" for the $k$-dimensional generalization of the now well-understood merging process in the plane. Intuitively, merging here means: take two subsets separated by a hyperplane and of about the same size, and combine them into a unique one. In the sequel, we simply assume that such a process is linear in the number of unfinished sites in the two subsets.

Please refer to Figs. 7 and 8 for the following result and its proof.

**Theorem 2.11.** *Let $S$ be a set of $N$ sites distributed in a $k$-dimensional unit hypercube $\mathcal{U}_k$, according to a quasi-uniform density probability $f$, with bounds $c_1$ and $c_2$ $(c_1 \leqslant c_2)$. Consider the following general scheme*:

**Divide step:** *divide $\mathcal{U}_k$ until reaching cells, each containing one single site, using a balanced k-d tree* [5];

**Merge step:** *then re-construct $\mathcal{U}_k$ through successive merges in reverse order* (*of the divisions*).

*If merging two subsets takes time proportional to the number of unfinished sites* (*with respect to $\mathcal{U}_k$*), *then the whole* merge *phase of the algorithm will be proportional to $N$.*

**Proof.** Using Theorem 2.10, the running time of the whole *merge* phase will be proportional to

$$\frac{N}{\sqrt[k]{N}}\,\frac{c_2}{\sqrt[k]{c_1}}\,\frac{k2^{k+1}\sqrt{\lceil (k+1)/2\rceil \lfloor (k+1)/2\rfloor}\sqrt[k]{k\Gamma(k/2+1)}}{\sqrt{\pi}}\chi, \tag{5}$$

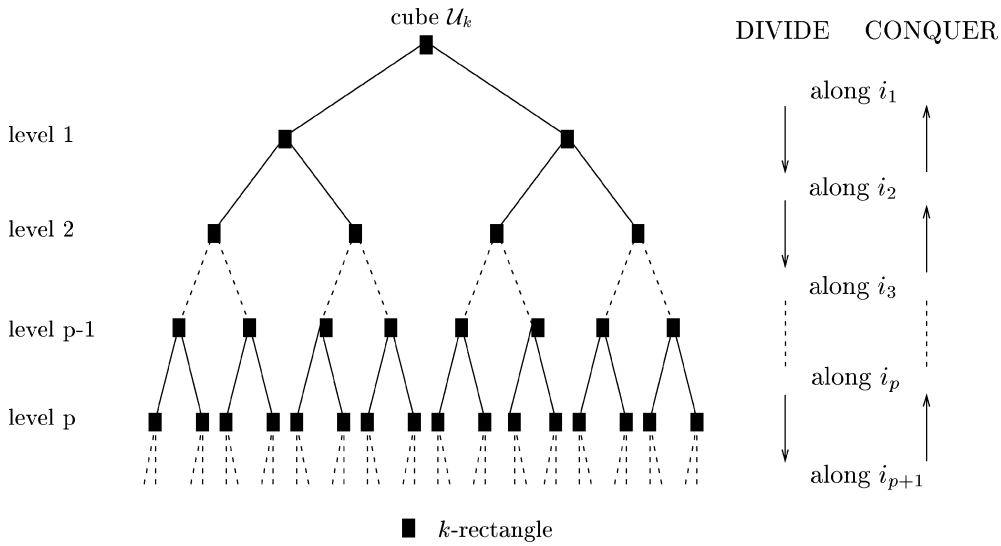


Fig. 7. Tree representation of divides and conquers according to a balanced k-d tree.

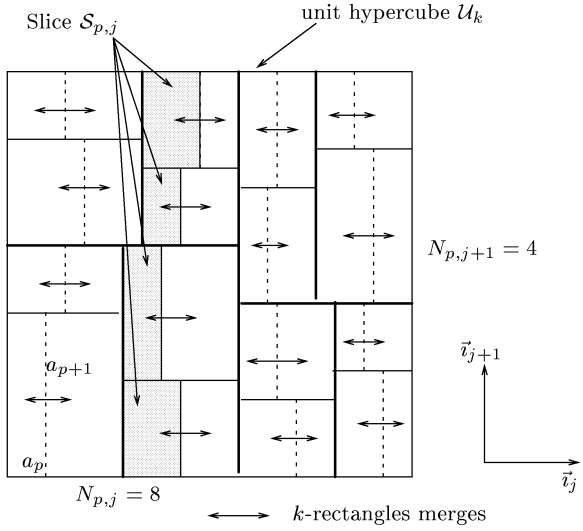


Fig. 8. Merges at level $p$: slice $\mathcal{S}_{p,j}$ with all its (shaded) hyperrectangle constituents.

where $\chi$ represents the sum of the surfaces of all $k$-rectangles involved in the merges. We shall decompose $\chi$ into sub-sums $\chi_p$, one for each merge level $p$ (Fig. 7).

Let $(\vec{\imath}_1, \ldots, \vec{\imath}_k)$ be an orthonormal basis, such that each axis is orthogonal to one hyperface of $\mathcal{U}_k$. The k-d tree produces a partition of $\mathcal{U}_k$ with $N$ $k$-rectangles.

**Definition 2.12.** Let us define slice $\mathcal{S}_{p,j}$ relative to direction $\vec{\imath}_j$ as one subset of the hyperrectangles partitioning $\mathcal{U}_k$ at merge $p$, such that the orthogonal projection of $\mathcal{S}_{p,j}$ on either hyperface of $\mathcal{U}_k$ orthogonal to $\vec{\imath}_j$ constitutes a partition of this hyperface (Fig. 8).

During the construction of the k-d tree, the division of space – orthogonally to one direction – doubles the number of slices relative to this direction. Moreover, note that the whole rectangular partition for $\mathcal{U}_k$ may be itself partitioned into slices with the same direction, whatever this direction. Let us call $N_{p,j}$ the number of slices relative to direction $\vec{\imath}_j$ at level $p$.

A simple induction argument yields

$$N_{p,j} = 2^{\lceil (p-j+1)/k \rceil},$$

for each merge level $p$, except the last – for which the right-hand side quantity is only an upper bound. Let us now evaluate $\chi_p$ by grouping $k$-rectangles into slices, and this, relatively to all directions. Since the surface of any hyperface of $\mathcal{U}_k$ is 1, the overall sum is equal to $2\sum_{j=1}^{k} N_{p,j}$, whence

$$\frac{1}{2}\chi_p \leqslant \sum_{j=1}^{k} 2^{\lceil (p-j+1)/k \rceil} \leqslant k 2^{\lceil p/k \rceil}.$$

Summing over all $h$ merge-levels yields

$$\frac{1}{2}\chi = \frac{1}{2}\sum_{p=1}^{h} \chi_p \leqslant \sum_{p=1}^{h} k 2^{\lceil p/k \rceil} = k \sum_{p=1}^{h} 2^{\lceil p/k \rceil} \leqslant k\left(k \sum_{w=1}^{\lceil h/k \rceil} 2^w\right) \leqslant k^2 2^{\lceil h/k \rceil + 1} \leqslant 4k^2 2^{\lceil h/k \rceil - 1}.$$

Since the division scheme is supported by a balanced k-d tree, its own height $h$ is such that $2^{h-1} < N \leqslant 2^h$, and hence

$$\sqrt[k]{N} > 2^{(h-1)/k} \geqslant 2^{\lceil h/k \rceil - 1} \Rightarrow \chi \leqslant 8k^2 \sqrt[k]{N}.$$

This implies that the average running time of the merge step is *linear* in the number of unfinished sites. It is, more precisely, proportional to

$$N \frac{c_2}{\sqrt[k]{c_1}} \frac{k^3 2^{k+4} \sqrt{\lceil (k+1)/2 \rceil \lfloor (k+1)/2 \rfloor} \sqrt[k]{k\Gamma(k/2+1)}}{\sqrt{\pi}}. \qquad \square$$

**Remark.** (1) The above proof shows more generally that the "conquer" phase of a $k$-dimensional divide-and-conquer algorithm is globally linear if merging two $k$-rectangles is proportional to $N^{(1-1/k)}$ times the sum of their surfaces.

(2) It would be very pleasant if one could use such a scheme to compute $k$-dimensional Delaunay triangulations. However, as of writing, there is no known algorithm for merging efficiently two Delaunay $k$-triangulations, with $k > 2$.

## 3. Application: 2D Delaunay triangulation in linear expected time after two-directional sorting

We now present the application of our results to the divide-and-conquer construction of the Delaunay triangulation in the plane, starting with a few definitions, and a commented overview of the algorithm.

### 3.1. Notations and definitions

In this section, let $\varepsilon$ be either 0 or 1, and $M_{[\varepsilon]}$ denote the corresponding coordinate of any point $M$ in the Euclidean plane $\mathbb{E}^2$ (0 $\rightsquigarrow$ abscissa, 1 $\rightsquigarrow$ ordinate). $<_{[\varepsilon]}$ will denote the lexicographical order relations between elements $M$ and $N$ of $\mathbb{E}^2$ defined as follows:

$$\begin{aligned} M <_{[0]} N \quad &\Longleftrightarrow \quad \big((M_{[0]} < N_{[0]}) \text{ or } (M_{[0]} = N_{[0]} \text{ and } M_{[1]} < N_{[1]})\big), \\ M <_{[1]} N \quad &\Longleftrightarrow \quad \big((M_{[1]} < N_{[1]}) \text{ or } (M_{[1]} = N_{[1]} \text{ and } M_{[0]} > N_{[0]})\big). \end{aligned} \tag{6}$$

If $A$ and $B$ are two subsets of $\mathbb{E}^2$, we shall write

$$A <_{[\varepsilon]} B \quad \Longleftrightarrow \quad \forall a \in A,\ b \in B, \quad a <_{[\varepsilon]} b.$$

If $|S|$ represents the number of elements in any set $S$ of $\mathbb{E}^2$, we shall say that subsets $S_1$ and $S_2$ form an $[\varepsilon]$-*division* of $S$ if and only if

(1) $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$,
(2) $|S_1| + |S_2| = |S|$, $0 \leqslant |S_1| - |S_2| \leqslant 1$,
(3) $S_1 <_{[\varepsilon]} S_2$.

In other words, $S$ is divided into two equally-sized subsets around the $[\varepsilon]$-median. (Recall that the median of a set $S$ for a given order relation may be defined as the element of $S$ with rank $\lfloor (|S|+1)/2 \rfloor$ in this order; hence, equally-sized should be understood as "with a maximum difference of 1 in the number of elements".) Fig. 9 illustrates typical $[\varepsilon]$-divisions.

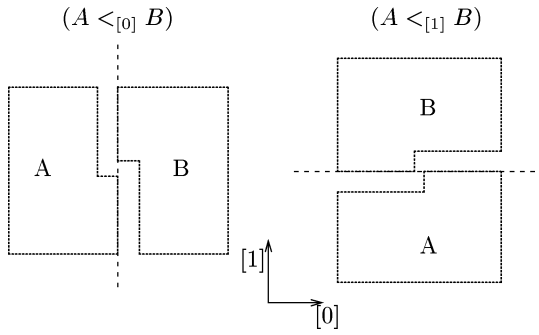


Fig. 9. [0]- and [1]-divisions.

### *3.2. Overview of the algorithm*

In the functional pseudo-code below, `t-set` stands for the data structure encapsulating the Delaunay triangulations returned by functions `Delaunay`, `Elementary_Delaunay` and `Merge`, and `v-set` is any appropriate organization of the data points (vertices), for instance list, array and so forth. Finally, note that function `Merge` accepts three arguments: the two triangulations resulting from recursion, and a direction.

At each recursion, function `Delaunay` is handed a couple $(S, \varepsilon)$, where $S$ is the current data (sub)set and $\varepsilon$ a division direction. In the initial call, $S$ is the whole set, and $\varepsilon$ is, say, 0 (i.e., the very first division is made along a vertical line through the $x$-median of the whole database).

```
Delaunay
output: t-set
input: v-set S, ε ∈ {0, 1}
local variables: v-set S1, S2
if (|S| ⩽ 1)
   return Elementary-Delaunay (S);
else
   (S1; S2) ← [ε]-division (S);
   return Merge (Delaunay (S1, 1 − ε), Delaunay (S2, 1 − ε), ε);
```

▷ *Divide*: If $S$ has at least two elements, it is $[\varepsilon]$-divided into $S_1$ and $S_2$, which are recursively triangulated using the orthogonal direction, $[1 - \varepsilon]$. Fig. 10 illustrates this principle: the root node of the recursion tree (top left) divides the whole set into two equally sized subsets on either side of the [0]-median. The two nodes at the next level divide each such 'half set' into two equally sized subsets below and above the [1]-median, and so forth until the bottom-most level (not shown), corresponding to elementary cells, containing single points.

In the sequel, we shall assume that the operation of $[\varepsilon]$-division is an O(1) process: this may simply be achieved by using an auxiliary linear space structure, say a 2-d tree, constructed in time $O(N \log N)$ during preprocessing, to store the nodes marking the successive divisions.

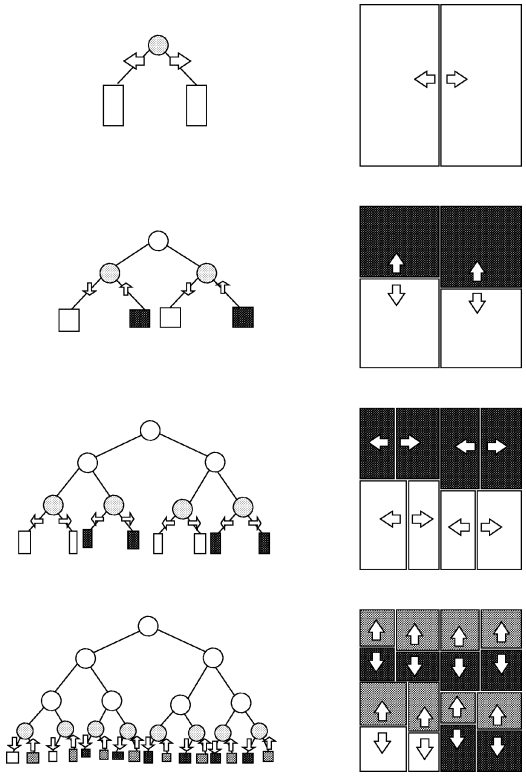

Fig. 10. The (initial steps of the) divide phase.

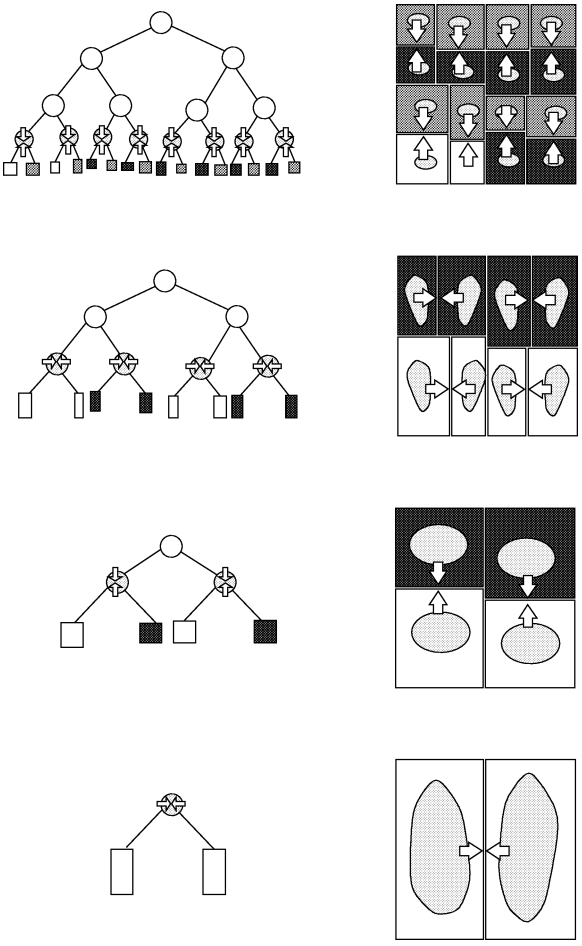

Fig. 11. The (final steps of the bottom-up) conquer phase.

▷ *Conquer*: The two resulting Delaunay triangulations are then "merged" into a single one, that is returned to the calling function. Note that the merge phase is akin to the one described in Lee and Schachter's algorithm [23]; however, the latter is essentially "one-directional", while the direction of the former is dictated by the value of $\varepsilon$ at each recursion level. In other words, to every $[\varepsilon]$-division corresponds an $[\varepsilon]$-merge. Fig. 11 illustrates this principle.

This is possible mainly because Lee and Schachter's (divide-and-)conquer technique only requires that the two sets to be merged be separable by a line, whatever its direction. As already noted by Edelsbrunner [15, p. 146], the merge is highly facilitated if we store, for each triangulation to be merged, the point closest to the separating line, according to the current $[\varepsilon]$-order. Such "cardinal points" are the (at most four) extrema for the two $<_{[\varepsilon]}$ order relations, and may be kept/retrieved with constant resources. Note that the definitions of the order relations (6) are slightly asymmetrical, to allow the same procedure to work in both vertical and horizontal directions. Practical considerations for implementing this two-directional merge function may be found in the first author's Ph.D. Dissertation [24].

### 3.3. Computational complexity

In this section, we consider the analysis of both the worst-case and average-case running times.

*Worst-case complexity.* Whatever method is chosen to perform the $[\varepsilon]$-divisions (preprocessing sort in two directions and construction of an auxiliary 2-d tree, or median extraction at each recursion), the overall running time of this process is $\Theta(N \log N)$. We also know [20] that the upper bound of the overall algorithm presented above is $O(N \log N)$, since merging two sub-triangulations takes linear time. Let us now show that the worst-case running time of the merge step is $\Omega(N \log N)$, using a method similar to the one devised by Katajainen and Koppinen [22] to compute the worst-case running time of their algorithm.

Consider the logarithmic spiral $\Gamma_a$ with equation $\rho = a\mathrm{e}^{m\theta}$ in polar coordinates, and note $M_i$ the point of the curve associated with $\theta_i$. This spiral has many properties, among which (Fig. 12) are the following:

(1) The tangent in $M$ makes a constant angle $V$ with line $OM(\tan V = 1/m)$.
(2) Let $\varpi$ be the center of curvature at point $M$. The circle through $M$ and centered in $\varpi$ contains the entire section of $\Gamma$ *before* $M$; the rest of the spiral *beyond* $M$ lies entirely outside this circle.
(3) The circle $C_{i,j,k}$ through three points $M_i$, $M_j$ and $M_k$ $(i < j < k)$ on the spiral contains in its interior all points of $\Gamma$ *before* $M_k$; the entire section of $\Gamma$ *beyond* $M_k$ lies outside $C_{i,j,k}$.
(4) The previous properties imply that constructing the Delaunay triangulation of a set $\mathcal{M}$ of sites $M_i$, $1 \leqslant i \leqslant N$, on a section of spiral, consists in linking $M_1$ to all other sites $M_j$ $(j \neq 1)$, then linking each site $M_i$, $1 \leqslant i < N$, to its successor $M_{i+1}$. Hence, if one adds to $\mathcal{M}$ a site from the spiral before $M_1$, the previous triangulation has no more valid triangle (Fig. 13).

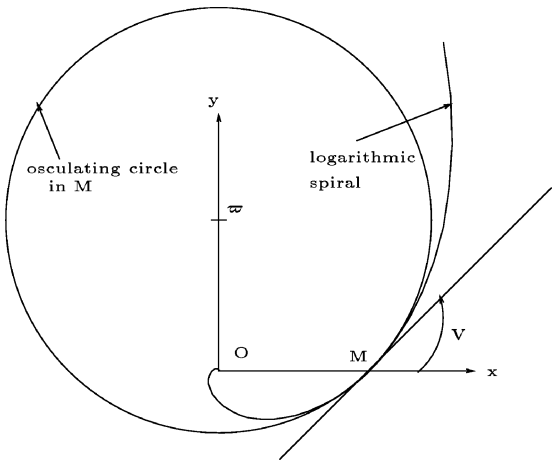


Fig. 12. Properties of the logarithmic spiral.

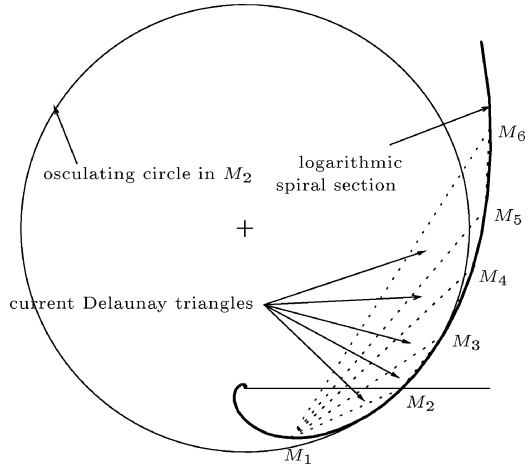


Fig. 13. Triangulation of a logarithmic spiral section: inserting a new site may cause the destruction of all already constructed Delaunay triangles.

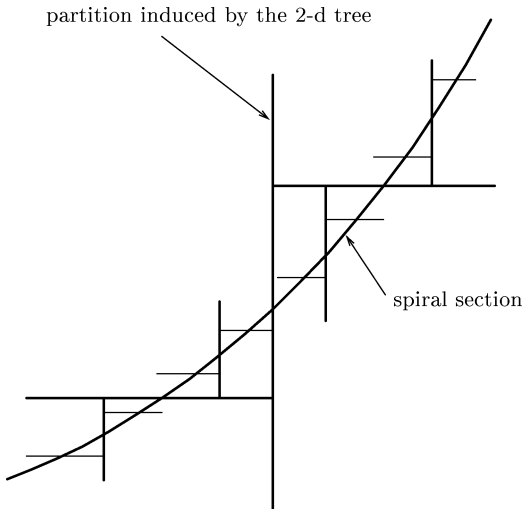


Fig. 14. Spiral partitioning induced by a 2-d tree: merging two adjacent cells may cause the destruction of all triangles in either.

Consider, for example, the logarithmic spiral $\Gamma_1$ with equation $\rho = \mathrm{e}^{\theta}$.

$$\begin{cases} x(\theta) = \mathrm{e}^{\theta}\cos\theta \\ y(\theta) = \mathrm{e}^{\theta}\sin\theta \end{cases} \quad\Rightarrow\quad \begin{cases} x'(\theta) = \sqrt{2}\,\mathrm{e}^{\theta}\cos\left(\theta + \frac{1}{4}\pi\right), \\ y'(\theta) = \sqrt{2}\,\mathrm{e}^{\theta}\sin\left(\theta + \frac{1}{4}\pi\right). \end{cases}$$

Consider the set $\{M_i\}_{i\in[1,N]}$ of sites with $-\pi/4 \leqslant \theta_1 < \cdots < \theta_N \leqslant \pi/4$, on $\Gamma_1$. Clearly, if $1 \leqslant i < j \leqslant N$,

$$M_i <_{[0]} M_j \quad \text{and} \quad M_i <_{[1]} M_j.$$

The 2-d tree will partition this set into cells, any two of which will be such that all the sites in the first are *before* those in the second (with respect to relations $<_{[0]}$ and $<_{[1]}$), or else *after* (Fig. 14). Hence, whenever a couple of such cells are to be merged, all the triangles from the first or the second must be destroyed. This implies the following recurrence equation:

$$T(N) = 2T\left(\frac{N}{2}\right) + \alpha N$$

for some strictly positive constant $\alpha$, which means that the algorithm has worst-case running time $\Omega(N\log N)$. Note that this lower bound may also be reached for sites on a conic, or various convex curves.

*Average-case complexity.* Let us suppose that sites are quasi-uniformly distributed over a unit square. First note that even if we use a linear expected time sorting algorithm [12] (which is possible thanks to the quasi-uniform distribution assumption), the construction of the 2-d tree remains $\Omega(N\log N)$. However, the expected running time of the conquer phase is improved. That the arguments of Section 2.6 hold for this algorithm is justified by the following theorem.

**Theorem 3.1.** *The worst-case complexity of the merge phase* (*in Lee and Schachter's algorithm*) *is bounded above by a linear function of the number of sites receiving new edges.*

This theorem – proven by Dwyer [13], using a previous result by Guibas and Stolfi [20] – implies that, in the case of a quasi-uniform distribution, the more general result of Theorem 2.11 yields an $O(N)$ expected running time for the entire merge phase, in a straightforward manner (the interested reader will find a specific analysis for the 2D case in [26]).

### *3.4. Results and comparison with other algorithms*

#### *3.4.1. Setting*

We conducted experimental comparisons between our 2-directional, divide-and-conquer 2-d tree based algorithm (nicknamed `2d` from here on) and the following divide-and-conquer algorithms for constructing 2-d Delaunay triangulations in the plane:

| | Author(s) | Merge type |
|---|---|---|
| `LS` | Lee and Schachter's [23] | 1-directional |
| `Dw` | Dwyer [13] | 2-directional, rows and then columns |
| `KK` | Katajainen and Koppinen [22] | 2-directional, quadtree order |
| `a2d` | Adam et al. [2] | 2-directional, (adaptive) 2-d tree order |

To make those tests as straightforward and objective as possible, the programs were all coded by the first author (at SETRA), using a uniform programming style. None was favored nor disadvantaged. In no case did we try to solve imprecision issues, as such was not the problem we were addressing. We simply used elementary techniques to prevent precision-related failures for all the algorithms (see also [37] for another robust method for constructing Voronoi or Delaunay diagrams in the plane):

- The point coordinates are stored using single precision, while computations are performed in double precision.
- A given determinant is reputed to be null if its computed value lies within $[-\varepsilon, +\varepsilon]$, where $\varepsilon$ is a data-dependent constant computed once and for all, given the maximum and minimum coordinates accepted as input.

A more sustained use of our implementations would have required well-known imprecision solving techniques (including the LEDA [10,28], LN [18] or CGAL libraries [17,30], or even LEA (the Lazy Exact Arithmetic library that the second author had taken part in the development of, at LISSE) [4,29]).

In addition to the actual running times, we have chosen to measure performance through the total number of edges created by the programs, since the running time for the triangulation (exclusive of sorting) is proportional to this quantity. This scheme frees interpretations from considerations on the speed of the processors, loading factors, and so forth. Extensive tests were conducted on sets of up to seven millions sites, uniformly distributed in a square domain. Both `LS` and `2d` were tested on a SUN workstation, on a Silicon Graphics Indy (200 MHz), and finally on a Convex C3 super-computer with 2 GB memory.

#### *3.4.2. Comparison with `LS`*

Fig. 15 illustrates the advantage of the bi-directional divide-and-conquer scheme over the one-directional one: `LS` creates many more initial triangles that are too thin to survive in the final Delaunay

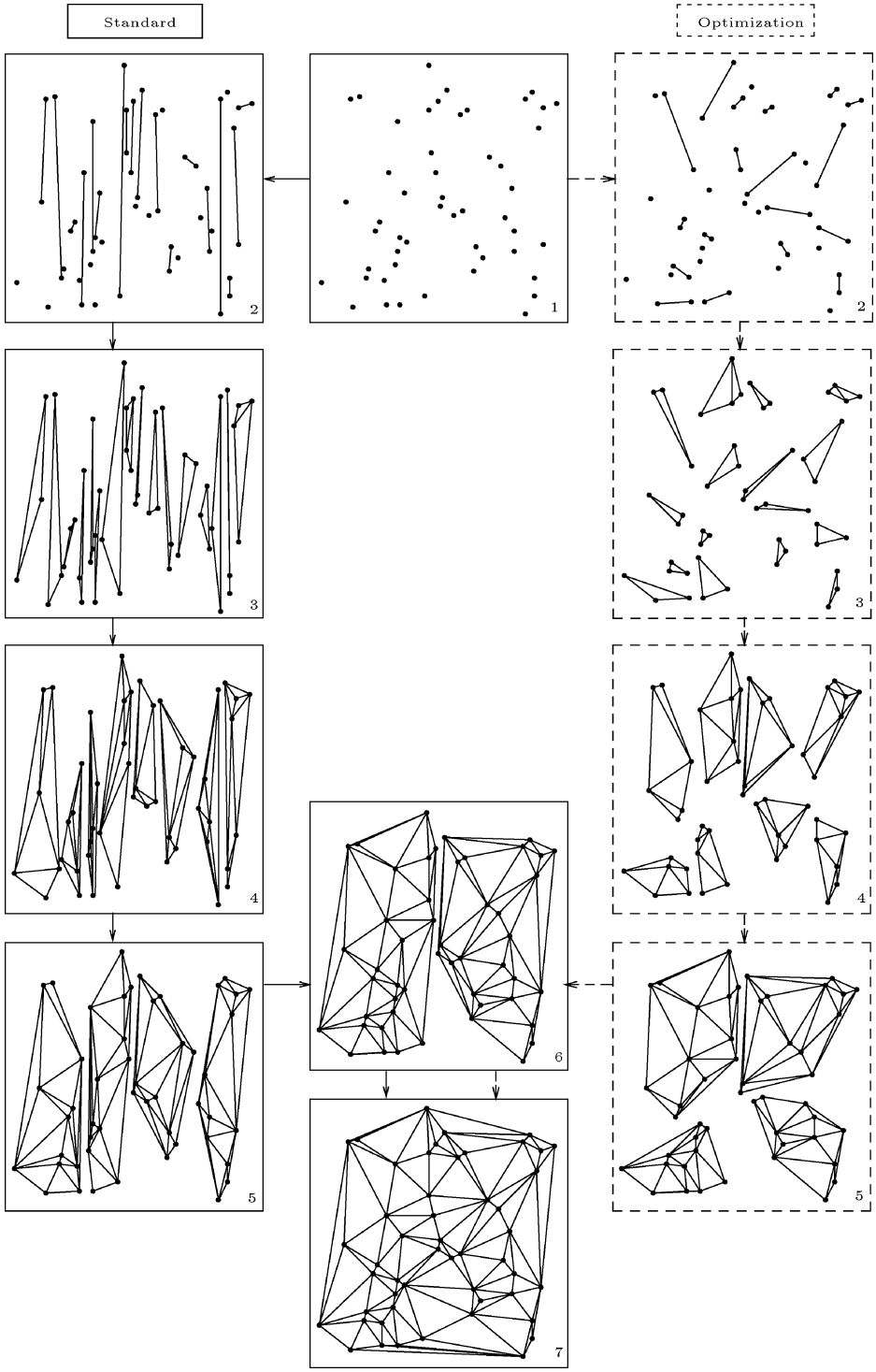


Fig. 15. Successive stages for LS (solid arrows) and 2d (dotted arrows) on 50 random sites.

| | Created | Destroyed |
|---|---|---|
| LS | $\sim \frac{1}{2}N \log N + N$ | $\sim \frac{1}{2}N \log N - 2N$ |
| 2d | $\sim 4N$ | $\sim N$ |

Fig. 16. Regressions on the average number of edges created and destroyed by LS and 2d for the triangulation of $N$ points.

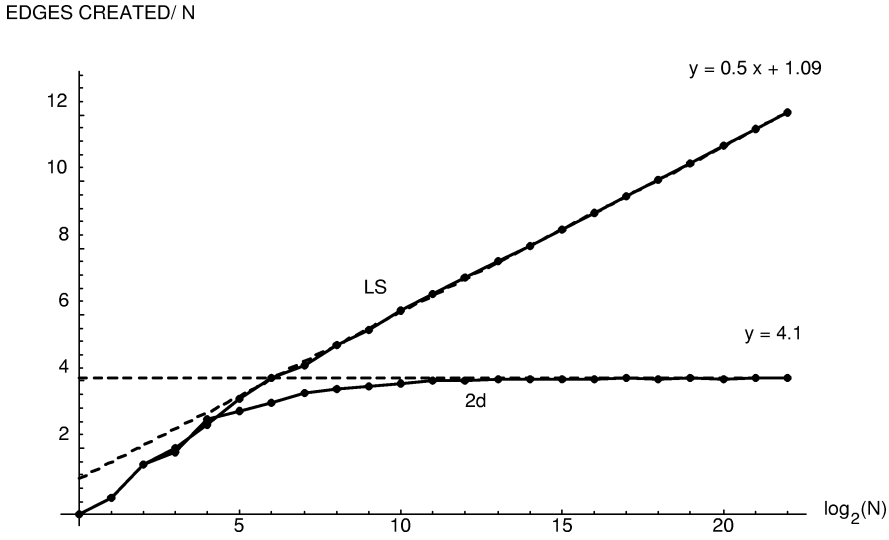


Fig. 17. Comparison between Lee and Schachter's algorithm (LS), and our optimized version (2d), for a uniform distribution.

triangulation than 2d does. The table in Fig. 16 illustrates the number of edges created and destroyed by LS and 2d for $N$ distinct sites uniformly distributed in a square, for a final total of about $3N$ edges in the triangulation. The statistics for a distribution in clusters were very similar, and are omitted. Fig. 17 illustrates the results obtained on the C3. These results strongly corroborate the theory, and show the asymptotic $\Theta(N \log N)$ and $\Theta(N)$ behaviour of LS and 2d, respectively (excluding preprocessing sorts). The difference between the running times of both versions becomes perceptible from 60 sites on, and, for 130,000 sites, 2d is already twice as fast as LS. The difference increases with $N$.

On a 200 MHz Indy workstation, 200,000 sites are triangulated in 7 seconds, after a sorting phase of 3 seconds. [2] The rate of triangulation (exclusive of sorting) is about 30,000 sites per second (between 50,000 and 60,000 triangles).

[2] Hence, although asymptotically slower than the merge phase, the preprocessing step is actually significantly faster in practice, as the operations involved in the latter are so much simpler than those involved in the former. We could check this phenomenon for 2d in all the tests that we conducted on either machine, even with the largest sets that could fit into the 2GB of internal memory.

*3.4.3. Comparison with KK, Dw and a2d*

According to Su and Drysdale [35,36], the reputedly best algorithms are KK and Dw, even for non uniform distributions. Our tests confirmed this fact for *uniform distributions* (see the top diagrams of Figs. 18 and 19). Note that 2d is only slightly slower. Both KK and Dw use hashing techniques, and it is well known that hashing performs best on uniform data and is faster than sorting. a2d [2] is based on an adaptive 2-d tree [19], in which the direction of division is chosen in function of the greatest dimension of the box bounding the sites. This algorithm is slightly slower because, although its merge phase is faster, the construction of the adaptive 2-d tree takes somewhat longer.

Now, for a *non-uniform distribution* with clusters (normal distribution $N(0, 0.01)$ at 10 randomly chosen sites in the unit square), 2d performs better than Dw and KK, the behaviours of which resemble that of LS. 2d is more appropriate for these distributions (see the bottom diagrams of Figs. 18 and 19). Let us also add that a2d creates, on average, even fewer edges than 2d, but that the overall running times are very close because the latter has a longer preprocessing step, as mentioned earlier. The interested reader will find other comparisons with other algorithms, and for other distribution types, in Lemaire's Ph.D. Dissertation [24].

*3.4.4. Shewchuk's implementation*

In [32], Shewchuk describes an idea similar to the one developed in this section (2d), and which led him to an implementation that may be found on the Internet. This research was conducted independently from ours, and Shewchuk does not analyze the performance of his method.

His program uses an exact adaptive arithmetic [33,34], allows the incremental insertion of constraint edges, the deletion of edges inside or outside a given boundary, the incremental insertion of new sites so the triangulation remains conforming. Shewchuk also shows experimentally that his algorithm is less prone to imprecision errors than Lee and Schachter's [23]. As a matter of fact, as the latter merges very thin and narrow strips, it must often determine the position of a site with respect to a segment, while the endpoints of the segments and the site are almost aligned. The error risk is hence much greater than is the case for an algorithm based on a 2-d tree (as Shewchuk's and ours), which forbids the major part of such very flat triangles (see Fig. 15).

## 4. Perspectives and conclusion

*Extension to other distributions.* It would be interesting to investigate the extensions to other distributions of the probabilistic results established in Section 2 in the case of quasi-uniform distributions.

*A conjecture on the linearity of a revisited version of Dwyer's algorithm.* In [13], Dwyer proposed a 2-d Euclidean Delaunay triangulation algorithm with $O(N \log\log N)$ average running time for $N$ sites uniformly distributed in a square. His algorithm uses a regular grid with $N/\log N$ cells, and may be summarized as follows:

(1) Triangulate each cell, with the help of Lee and Schachter's algorithm [23].
(2) Triangulate each row by merging pairs of adjacent cells.
(3) Merge rows by pairs until the triangulation of the set is completed.

We conjecture that, by using a grid with $N$ cells, this algorithm has a linear expected running time under the same assumptions. The first stage is linear on average, since the number of sites in a cell is constant,

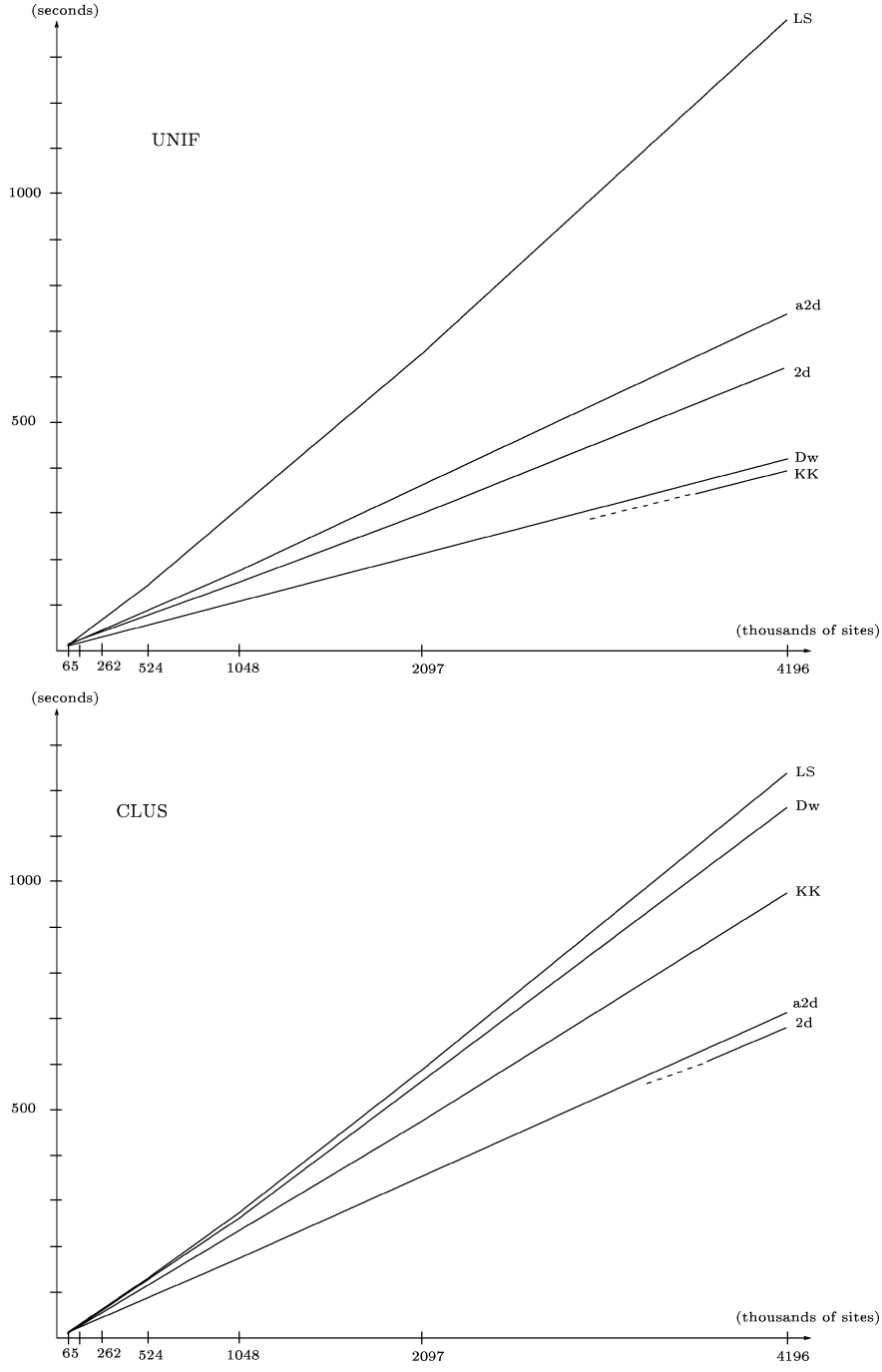


Fig. 18. Comparison of the whole running time of Delaunay triangulation algorithms (including preprocessing time): the algorithms are tested on a Convex C3 – a super-computer with 2 GB of memory – on data sets of up to seven millions sites distributed uniformly (top diagram) or non-uniformly with clusters (bottom diagram) in a square domain.

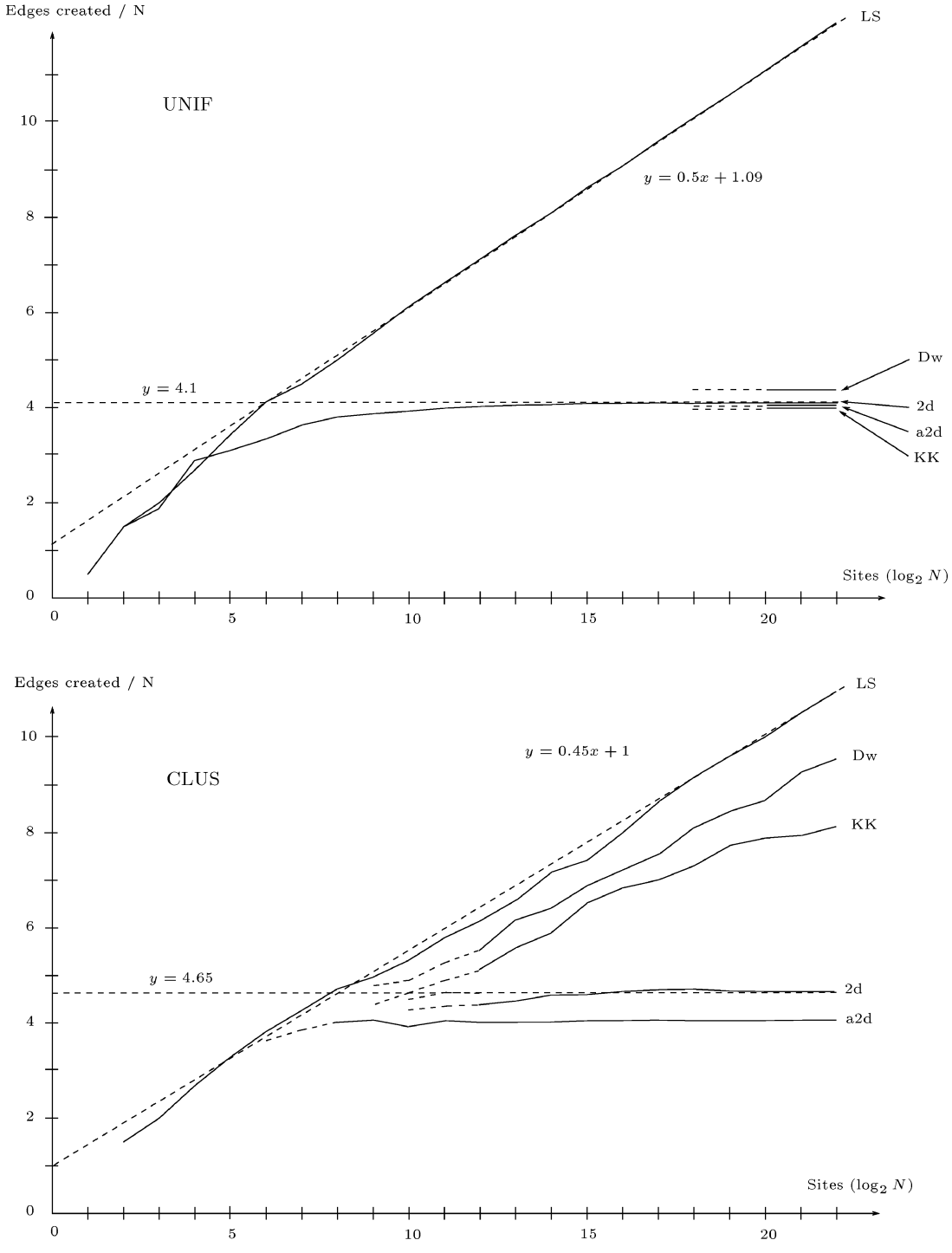


Fig. 19. Comparison of Delaunay triangulation algorithms according the total number of edges created, in the same conditions as for the previous figure.

on average. Using Theorem 2.10, it is possible to show that stage 3 is linear on average. However, this theorem does not allow to prove the expected linearity of stage 2 (triangulation of a row). There is a strong experimental evidence that the triangulation phase of rows (containing up to 8 million sites) is linear. The rigorous proof of such a conjecture remains an open problem.

*Delaunay triangulation in any dimension.* We have seen that the idea presented in this paper could be applied to the two-dimensional case with an effective speed-up of Lee and Schachter's optimal algorithm, while preserving the worst-case optimality. In higher dimensions, the difficulty resides in the merging of sub-triangulations. In the literature, the merge process is often described in dimension 2, very rarely in dimension 3, and never in higher dimensions. Cignoni et al. [11] present a *divide-and-conquer* algorithm in any dimension, but the merge phase is incremental, and the problem of merging in any dimension is not solved.

However, here are a few arguments that prove that there still is room for hope:

(1) The Ph.D. Dissertations of Elbaz [16] and Adam [1] present a *divide-and-conquer* algorithm for constructing Delaunay triangulations in 3-space. The process of merging sub-triangulations seems to be correctly described; however, the expected running time analysis is not done. We hope that the results presented in this paper may help in that respect.
(2) Buckley [9] suggests a *divide-and-conquer* algorithm to construct convex hulls in dimension 4, which the author generalizes to any dimension. Recall the strong relation between convex hulls and Delaunay triangulations: computing a Delaunay triangulation in dimension $k$ may be reduced to computing a convex hull in $(k+1)$-space; to that effect, it suffices to project the sites on a paraboloid with axis orthogonal to the hyperplane containing them, to compute the convex hull of these projected sites, and to project it on the initial hyperplane, which yields the sought triangulation.
(3) Merging sub-triangulations in dimension greater than two has never been proven impossible. It is quite probable that many researchers have only been intimated by the complexity of the enterprise.

This paper is intended to give a new motivation for the design of an algorithm to merge sub-triangulations in any dimension, that would allow the construction of multi-dimensional Delaunay triangulations in a divide-and-conquer fashion.

*By way of conclusion.* The evaluation of the expected number of unfinished sites in a $k$-rectangle, under quasi-uniform distribution, is an interesting result in itself, which may be used to analyze various algorithms, that must, however, be more or less related to Delaunay triangulations, since the notion of "unfinished sites" only has a meaning in such a context. It may be added to the probabilistic results already obtained by Bern et al. [8] on the Delaunay triangulation in any dimension.

The Delaunay triangulation algorithm presented in this paper has a provably good behaviour for uniform distributions and seems to be the most efficient for cluster distributions. One of our future research goals is to prove that this method may be generalized in a $k$-dimensional setting, to yield, after multi-dimensional pre-sorting, a Delaunay triangulation algorithm, the expected running time of which would be proportional to the number of sites. We have shown that, although there is no known algorithm for efficiently merging two Delaunay sub-triangulations in $k$-space, the expected running time of the construction is *potentially* proportional to the number of points, provided they are quasi-uniformly distributed in a hypercube. On the basis of these results, the chances are great that such an algorithm would be efficient, and might compete with Dwyer's incremental algorithm [14], which uses a pre-

partitioning of the sites by means of a regular grid. However, we are fully aware that merging sub-triangulations in higher dimensions is a tantalizing task!

**Acknowledgements**

The authors wish to thank Olivier Devillers (INRIA, Sophia Antipolis, France) for his useful comments on an early version of this paper, Luc Devroye (McGill University, Montreal, Canada) for his help, encouragements and many comments, among which the remark in the appendix, Professor Kurt Mehlhorn for his patience, perspicacity and kind help, and an anonymous referee for many judicious remarks and pieces of advice that greatly helped improve an earlier version of this paper.

## Appendix A. A technical lemma for the proof of Theorem 2.10

We now give a justification of the bound used in the course of proving Theorem 2.10 of Section 2.5.

**Lemma A.1.** *Let* $\mathcal{I}_{n,k} = \int_0^1 (1-x^k)^n \,\mathrm{d}x$ *with integers* $n \geqslant 1$ *and* $k \geqslant 2$. *Then*

$$\mathcal{I}_{n,k} < \frac{1}{\sqrt[k]{n+1}}.$$

**Proof.** We shall argue by induction on $n$, for fixed $k \geqslant 2$.

*Preliminary step.* Let us first derive the recurrence equation

$$\mathcal{I}_{n,k} = \frac{kn}{kn+1} \cdot \mathcal{I}_{n-1,k}, \tag{A.1}$$

using the deep relation between $\mathcal{I}_{n,k}$ and the beta function:[3] setting $u = x^k$ in the integral defining $\mathcal{I}_{n,k}$, we may write

$$\mathcal{I}_{n,k} = \frac{1}{k} \int_0^1 u^{1/k-1}(1-u)^{(n+1)-1} \,\mathrm{d}u = \frac{1}{k}\mathcal{B}\left(\frac{1}{k}, n+1\right) = \frac{1}{k} \cdot \frac{\Gamma(1/k)\Gamma(n+1)}{\Gamma(1/k+n+1)}.$$

This yields

[3] The beta function is defined, for all strictly positive real numbers $m, n$ as

$$\mathcal{B}(m,n) = \int_0^1 x^{m-1}(1-x)^{n-1} \,\mathrm{d}x = \frac{\Gamma(m)\Gamma(n)}{\Gamma(m+n)},$$

where $\Gamma$ stands for the gamma function, defined for any strictly positive real number $n$ as

$$\Gamma(n) = \int_0^\infty x^{n-1}\mathrm{e}^{-x} \,\mathrm{d}x.$$

$$\begin{aligned}\frac{\mathcal{I}_{n,k}}{\mathcal{I}_{n-1,k}} &= \frac{\Gamma(1/k)\cdot\Gamma(n+1)}{\Gamma(1/k+n+1)}\cdot\frac{\Gamma(1/k+n)}{\Gamma(1/k)\cdot\Gamma(n)}\\ &= \frac{n\Gamma(n)}{\Gamma(n)}\cdot\frac{\Gamma(1/k+n)}{(1/k+n)\Gamma(1/k+n)} \quad (\text{since } \Gamma(t+1)=t\Gamma(t),\ \forall t>0)\\ &= n\cdot\frac{1}{1/k+n} = \frac{kn}{kn+1}.\end{aligned}$$

*Inductive argument.* Recall that $k$ is a fixed integer (strictly greater than 1); let $\mathbb{P}(n)$ denote the inequality we wish to prove (i.e., $\mathbb{P}(n)$: $\mathcal{I}_{n,k} < 1/\sqrt[k]{n+1}$). As

$$\forall k \geqslant 2, \quad (k+1)^k = k^k + kk^{k-1} + \cdots > 2\cdot k^k \Rightarrow \frac{k}{k+1} < \frac{1}{\sqrt[k]{2}},$$

$\mathbb{P}(1)$ is clearly true.

*Induction hypothesis*: Suppose $n \geqslant 2$ is some integer such that $\mathbb{P}(n-1)$ is true, i.e.,

$$\mathcal{I}_{n-1,k} < \frac{1}{\sqrt[k]{n}}.$$

We now show that

$$\mathbb{P}(n-1) \Rightarrow \mathbb{P}(n),$$

starting with a classical binomial expansion and minorization,

$$(kn+1)^k = (kn)^k + k\cdot(kn)^{k-1}\cdot 1 + \cdots > k^k n^k + k^k n^{k-1} = (kn)^k\left(\frac{n+1}{n}\right).$$

Hence,

$$\frac{1}{\sqrt[k]{n+1}} > \frac{kn}{kn+1}\cdot\frac{1}{\sqrt[k]{n}} > \frac{kn}{kn+1}\cdot\mathcal{I}_{n-1,k} = \mathcal{I}_{n,k},$$

using the induction hypothesis and relation (A.1), successively. □

**Remark** (Luc Devroye). The convexity of $x \rightsquigarrow \log\Gamma(x)$ yields

$$\begin{aligned}\log\left(\Gamma\left(n+1+\frac{1}{k}\right)\right) &\leqslant \log\bigl(\Gamma(n+1)\bigr) + \frac{\log(\Gamma(n+2)) - \log(\Gamma(n+1))}{(n+2)-(n+1)}\cdot\frac{1}{k}\\ &= \log\bigl(\Gamma(n+1)\bigr) + \frac{1}{k}\log(n+1),\end{aligned}$$

which allows to write

$$\frac{\Gamma(n+1)}{\Gamma(n+1+1/k)} \geqslant \frac{1}{\sqrt[k]{n+1}}.$$

Hence, $\mathcal{I}_{n,k}$ differs from the upper bound we provide ($(n+1)^{-1/k}$) by a factor of at most $1/\Gamma(1+1/k)$ (the bound is reached for $k=1$ and $k\to+\infty$). For $k$ in the range $[1,+\infty[$, $1 \leqslant 1/\Gamma(1+1/k) < 1.15$, and the difference between $I_{n,k}$ and its majorization $1/\sqrt[k]{n+1}$ never exceeds 15%.

## References


[1] B. Adam, Construction des diagrammes de Delaunay dans le plan et dans l'espace, Ph.D. Thesis, Université de Haute-Alsace, France, 1996.

[2] B. Adam, M. Elbaz, J.C. Spehner, Construction du diagramme de Delaunay dans le plan en utilisant les mélanges de tris, in: Quatrièmes Journées de l'AFIG, Dijon, France, 1996, pp. 215–223.

[3] F. Aurenhammer, Voronoi diagrams – a survey of a fundamental geometric data structure, ACM Comput. Surveys 23 (3) (1991) 345–405.

[4] M.O. Benouamer, P. Jaillon, D. Michelucci, J.-M. Moreau, A lazy arithmetic library, in: Proc. of 11th IEEE Symposium on Computer Arithmetic, Windsor, Ontario, 1993, pp. 242–249.

[5] J.L. Bentley, Multi-dimensional binary search trees used for associative searching, Comm. ACM 18 (9) (1975) 509–517.

[6] M. Berger, Géométrie, 3. Convexes et polytopes polyèdres réguliers, aires et volumes, Fernand Nathan, Paris, 1974.

[7] M. Berger, Geometry, Vols. 1–2, Springer, Berlin, 1987.

[8] M. Bern, D. Eppstein, F. Yao, The expected extremes in a Delaunay triangulation, Internat. J. Comput. Geom. Appl. 1 (1991) 79–91.

[9] C.E. Buckley, A divide-and-conquer algorithm for computing 4-dimensional convex hulls, in: Proc. of Computational Geometry and its Applications, Lecture Notes in Computer Science, Vol. 333, Springer, Berlin, 1988, pp. 113–135.

[10] C. Burnikel, J. Könnemann, K. Mehlhorn, S. Näher, S. Schirra, C. Uhrig, Exact geometric computation in LEDA, in: Proc. of 11th Annu. ACM Symposium on Computational Geometry, 1995, pp. C18–C19.

[11] P. Cignoni, C. Montani, R. Scopigno, Dewall: a fast divide-and-conquer Delaunay triangulation algorithm in $E^d$, Technical Report 95-22, Instituto CNUCE-CNR, Pisa, Italy, 1995.

[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press/McGraw-Hill, Cambridge, MA, 1990.

[13] R.A. Dwyer, A faster divide-and-conquer algorithm for constructing Delaunay triangulations, Algorithmica 2 (1987) 137–151.

[14] R.A. Dwyer, Higher-dimensional Voronoi diagrams in linear expected time, Discrete Comput. Geom. 6 (1991) 343–367.

[15] H. Edelsbrunner, Algorithms in Combinatorial Geometry, EATCS Monographs on Theoretical Computer Science, Vol. 10, Springer, Heidelberg, Germany, 1987.

[16] M. Elbaz, Les diagrammes de Voronoi et de Delaunay dans le plan et dans l'espace, Ph.D. Thesis, Université de Haute-Alsace, France, 1992.

[17] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, S. Schoenherr, On the design of CGAL, the computational geometry algorithms library, Technical Report MPI-I-98-007, Max Plank Institute for Computer Science, 1998.

[18] S. Fortune, C. van Wyk, Efficient exact arithmetic for computational geometry, in: Proc. of 9th ACM Symposium on Computational Geometry, San Diego, CA, 1993, pp. 163–172.

[19] J.H. Friedman, J.L. Bentley, R.A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Trans. Math. Software 3 (1977) 209–226.

[20] L.J. Guibas, J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, ACM Trans. Graphics 4 (1985) 74–123.

[21] M. Iri, T. Ohya, K. Murota, Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms, J. Oper. Res. Soc. Japan 27 (1984) 306–337.

[22] J. Katajainen, M. Koppinen, Constructing Delaunay triangulations by merging buckets in quadtree order, Ann. Soc. Math. Polon. Ser. IV Fund. Inform. 11 (3) (1988) 275–288.

[23] D.T. Lee, B.J. Schachter, Two algorithms for constructing a Delaunay triangulation, Internat. J. Comput. Inform. Sci. 9 (1980) 219–242.
[24] C. Lemaire, Triangulation de Delaunay et arbres multidimensionnels, Ph.D. Thesis, ENSM.SE, Saint-Étienne, France, 1997 (in French). Postscript version available at: http://lisse.emse.fr.
[25] C. Lemaire, J.-M. Moreau, Triangulation de Delaunay euclidienne dans le plan: optimisation du divide and conquer à l'aide d'un tri selon deux directions, Technical Report 02D07669, SETRA, Bagneux, France, 1996.
[26] C. Lemaire, J.-M. Moreau, Amélioration de la complexité en moyenne de l'algorithme de Lee et Schachter par division et fusion bi-directionnelles, Revue Internationale de CFAO et d'Informatique Graphique 12 (4) (1997) 317–336.
[27] A. Maus, Delaunay triangulation and the convex hull of $n$ points in expected linear time, BIT 24 (1984) 151–163.
[28] K. Mehlhorn, S. Näher, LEDA: A Platform for Combinatorial and Geometric Computing, Cambridge University Press, New York, 1999.
[29] D. Michelucci, J.-M. Moreau, Lazy arithmetic, IEEE Trans. Comput. 46 (9) (1997) 961–975.
[30] M.H. Overmars, Designing the computational geometry algorithms library CGAL, in: Proc. of 1st ACM Workshop on Applied and Computational Geometry, Lecture Notes in Computer Science, Vol. 1148, Springer, Berlin, 1996, pp. 113–119.
[31] M.I. Shamos, D. Hoey, Closest-point problems, in: Proc. of 16th Annu. IEEE Symposium on Foundations of Computer Science, 1975, pp. 151–162.
[32] J.R. Shewchuck, Triangle: engineering a 2d quality mesh generator and Delaunay triangulator, in: M.C. Lin, D. Manocha (Eds.), Proc. of 1st ACM Workshop on Applied and Computational Geometry, Lecture Notes in Computer Science, Vol. 1148, Springer, Berlin, 1996, pp. 124–133. Program available at http://www.cs.cmu.edu/~quake/triangle.html.
[33] J.R. Shewchuk, Robust adaptive floating-point geometric predicates, in: Proc. of 12th ACM Annu. Symposium on Computational Geometry, 1996, pp. 141–150.
[34] J.R. Shewchuk, Adaptive precision floating-point arithmetic and fast robust geometric predicates, Technical Report CMU-CS-96-140, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, May 1996.
[35] P. Su, R.L.S. Drysdale, A comparison of sequential Delaunay triangulation algorithms, in: Proc. of 11th Annu. ACM Symposium on Computational Geometry, 1995, pp. 61–70.
[36] P. Su, R.L.S. Drysdale, A comparison of sequential Delaunay triangulation algorithms, Computational Geometry 7 (1997) 361–385.
[37] K. Sugihara, M. Iri, Construction of the Voronoi diagram for 'one million' generators in single-precision arithmetic, Proc. of the IEEE 80 (9) (1992) 1471–1484.