Average-case analysis of a class of divide-and-merge algorithms, with an application to the $2d$ Delaunay triangulation

C. Lemaire SETRA[1]

J-M. Moreau LISSE[2]

December, 1997

Technical report **RR 97.11**

[1]Adress: 46 av. A. Briand, BP 100, 92225 Bagneux Cedex (France). Email: `lemaire@setra.fr`

[2]Address: 158, cours Fauriel, F 42 023 Saint-Étienne. Email: `moreau@emse.fr`

Abstract

This paper exploits the notion of "unfinished site", introduced by Katajainen and Koppinen [18] in the analysis of a two-dimensional Delaunay triangulation algorithm, based on a regular grid. We extend their proof, improve their bounds and, more important, we generalize the notion and its properties to any dimension $k \geq 2$: 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-merge schemes based on balanced k-d trees.

This general result is then applied to the analysis of an algorithm for constructing Delaunay triangulations in the plane ([22, 21]), that is shown, in the first author's Ph.D. Thesis ([20]), to be one of the most efficient.

Keywords: Delaunay triangulation, k-d tree, worst-case complexity, average-case complexity, multi-dimensional divide-and-conquer.

Résumé

Cet article exploîte la notion of "site inachevé", introduite par Katajainen and Koppinen [18] dans l'analyse d'un algorithme de triangulation de Delaunay en deux dimensions, fondé sur sune grille régulière. Nous étendons leur preuve, et surtout nous généralisonts la notion et ses propriétés à toute dimension $k \geq 2$: dans le cas d'une distribution uniforme, le nombre moyen de sites inachevés dans un k-rectangle est $O(N^{1-1/k})$. Ceci implique, sous certaines conditions, la linéarité d'une classe d'algorithmes de type diviser-pour-résoudre fondés sur des k-d arborescences équilibrées.

Ce résultat général est ensuite appliqué à l'analyse d'un algorithme pour construire la triangulation de Delaunay dans le plan ([22, 21]), dont le premier auteur montre dans sa thèse de doctorat ([20]), qu'il est l'un des plus efficaces connus.

Contents

1

Introduction

The Voronoi diagram of a set of N distinct points in the plane is a well-studied problem ([2]), for which Shamos & Hoey published the first optimal *divide-and-conquer* solution ([26]). 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 & Schachter ([19]), to be later refined by Guibas & Stolfi ([16]).

In 1984, Ohya, Iri and Murota ([32]) 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, Iri and Murota themselves ([32]), and A. Maus ([23]) 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, R. Dwyer published a $[O(N \log\log N), O(N \log N)]$ expected / worst-case variant of Lee & Schachter's algorithm, in which the domain is partitioned into $N/\log N$ cells, that are triangulated, merged into rows, and rows are eventually merged together. In their 1988 paper, [18], J. Katajainen and M. Koppinen modified Dwyer's algorithm (partition into N cells, merge of cells in a quadtree-like order) to achieve linear expected time. Their average-case analysis exploited the notion of *unfinished sites* in a rectangular domain. Finally, R. Dwyer published the first k-dimensional method with expected linear behaviour in 1991 ([11]), also based on buckets.

We now go one step further in this progression: after stating some definitions and general assumptions on the distribution of sites (Part 2: Section 2.1), we generalize the notion of unfinished site to any dimension (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 2.4), 2.5. The upper bounds we provide are valid in any dimension, and, when applied to the $2d$ case, improve upon those established by Katajainen and Koppinen.

[. . . / . . .]

Moreover, we modify Katajainen and Koppinen's proof and extend it to any dimension. These results may be used to analyze the running time of any k-dimensional divide-and-conquer algorithm for which the notion of unfinished site has a meaning, and hence is more or less directly related to the Delaunay triangulation.

Next (Part 3), we present a divide-and-merge scheme based on a (balanced) k-d tree [4], and show that the expected running time of the multi-dimensional merge phase is linear in the number of unifinished sites, in the case of quasi-uniform distribution in a hypercube. It is hoped that this result may be used in the design (and analysis) of an algorithm for constructing Delaunay triangulations in any dimension, using a divide-and-conquer scheme based on a k-d tree.

This general result is then used (Section 3.2) to analyze a new algorithm to construct the Delaunay triangulation in the plane [22, 21], that is shown, in the first author's Ph.D. Thesis [20], to be among the most efficient. Section 4 then gives experimental results on this algorithm , and a comparison with the best known algorithms, as established by Su and Drysdale [30, 31].

The paper closes on prospects about possible divide-and-conquer algorithms for constructing Delaunay triangulations in k-space (Section 5), and open problems.

2

Mathematical preliminaries

2.1 Quasi-uniform assumption

Throughout this paper, we shall consider sets of N points (also called *sites*) in a Euclidian space of dimension k, 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 $c_1 \leq c_2$, and of a probability density f such that:

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

Thus, 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

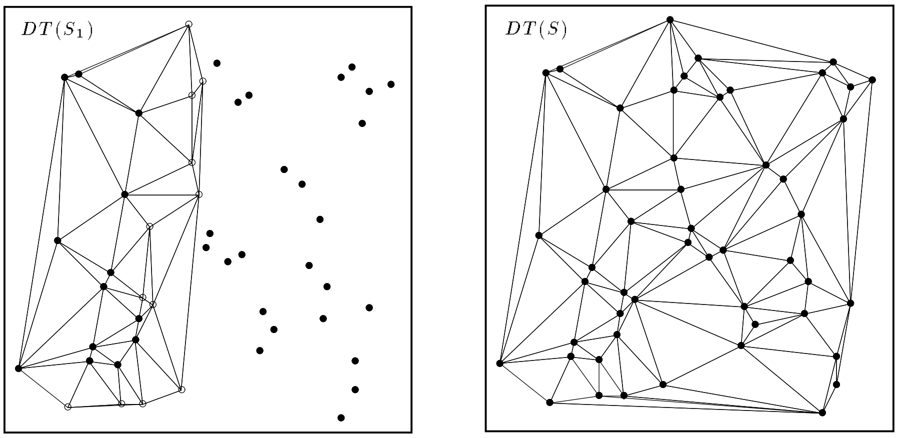
The notion of *unfinished site* was introduced by Katajainen and Koppinen [18] for the analysis of an algorithm to construct Delaunay triangulations in the plane, that partitions the domain according to a regular grid.

Consider, for instance, the planar Delaunay triangulation, $DT(S_1)$, of the point set S_1, subset of S (refer to Figure 2.1). 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 following paragraphs 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. We shall denote by $DT(S)$ 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.*



• *Finished* site in $DT(S_1)$ wrt $DT(S)$.

∘ *Unfinished* site in $DT(S_1)$ wrt $DT(S)$

Figure 2.1: 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 $T(S_1)$ wrt $T(S_2)$ 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 $T(S_1)$ wrt $T(S_2)$).*

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

Proof Let (s,p_1),… (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),… (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, $(s,r) \setminus \{s\}$ 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 2.0.1 *Let $T(S_1) <_\Delta T(S_2) <_\Delta T(S_3)$ be three triangulations. A site $s \in S_1$ is finished in $T(S_1)$ wrt $T(S_3)$ if and only if it is both finished in $T(S_1)$ wrt $T(S_2)$, and in $T(S_2)$ wrt $T(S_3)$.*

Proof The final condition implies the previous one as consequence of Definition 2.2.
Conversely, suppose s is finished in $T(S_1)$ *wrt* $T(S_3)$. If (s,q) is one edge in $T(S_3)$, $q \in S_1$; since $q \in S_2$, and, using Proposition 2.1, we may conclude that s is finished in $T(S_2)$ *wrt* $T(S_3)$. Finally, because of Definition 2.2, s is also finished in $T(S_1)$ *wrt* $T(S_2)$. □

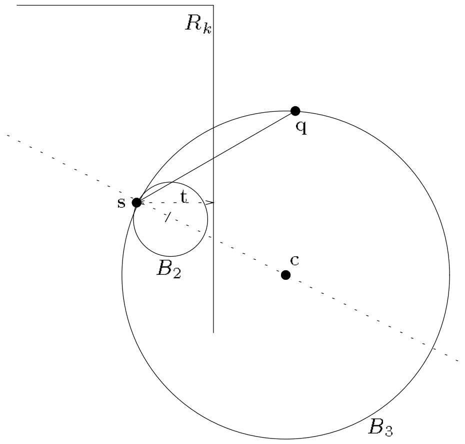


Figure 2.2: Ball $\mathcal{B}_2$ has no sites in its interior.

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 2.1 *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 $\frac{t}{2}$, centered at distance $\frac{t}{2}$ from s, with no sites in its interior.*

Proof Referring to Figure 2.2, if s is an unfinished site in k-rectangle $\mathcal{R}_k$, there is a site q belonging to $\mathcal{U}_k \setminus \mathcal{R}_k$ 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 $\mathcal{B}_2$ the k-ball with center lying on segment $[sc]$ at distance $\frac{t}{2}$ from s. $\mathcal{B}_2$ is contained in $\mathcal{B}_3$, hence $\mathcal{B}_2$ has no sites in its interior. $\square$

Corollary 2.0.2 *In the setting of Lemma 2.1, 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 wirth apex s, symmetry axis sp, and angle $\theta = \arccos \frac{r_1}{t}$. Then, $\mathcal{C}_{\mathcal{B}_1} = \mathcal{C} \cap \mathcal{B}_1$ is empty of sites (cf. Figure 2.3).*

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 θ be the angle formed by segments $[so]$ and $[sm]$.

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

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. $\square$

Lemma 2.2 *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.*

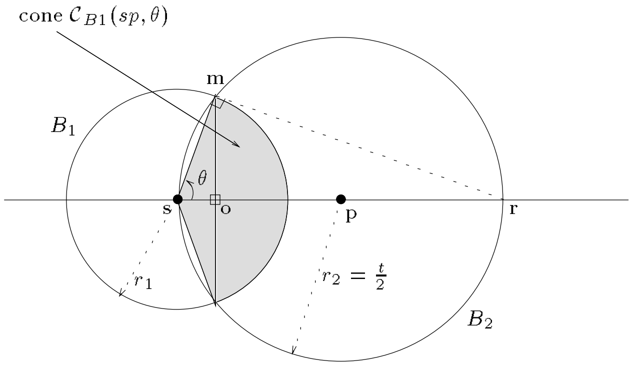


Figure 2.3: Cone $\mathcal{C}_{B1}(sp,\theta)$ has no sites in its interior.

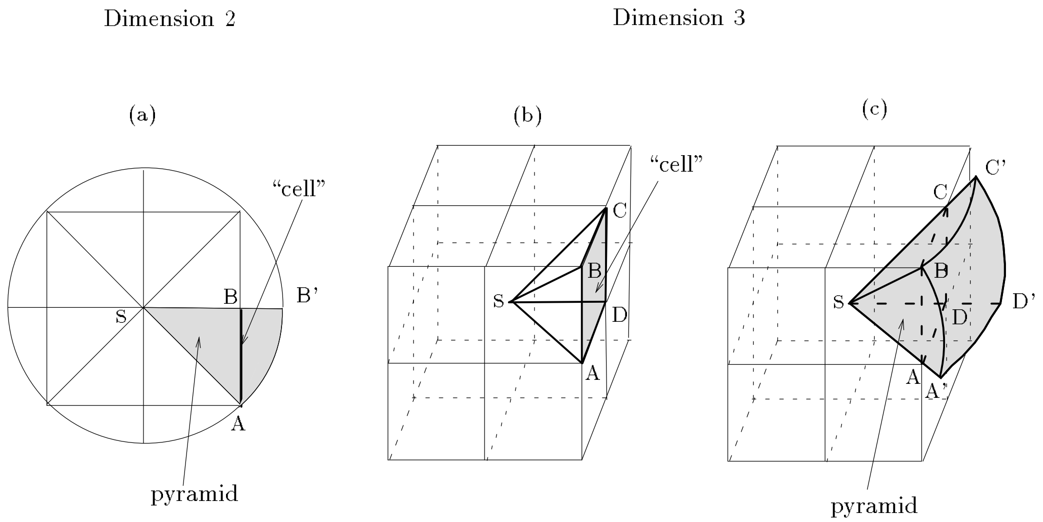


Figure 2.4: Decomposition of a ball into pyramids.

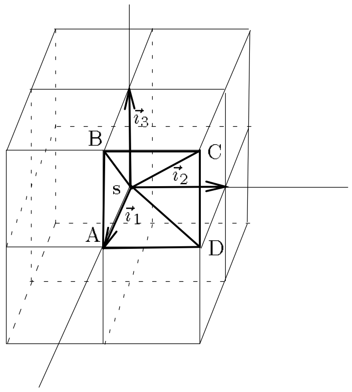
2. $\forall M_1, M_2 \in \mathcal{P} \setminus \{s\},\ \cos(\widehat{M_1 s M_2}) \geq (\lceil \frac{k+1}{2} \rceil \lfloor \frac{k+1}{2} \rfloor)^{-1/2}$

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 [5, 6]. 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:

$$\mathcal{V}(\mathcal{P}) = \frac{\mathcal{V}(\mathcal{B}_1)}{k2^k} = \frac{\pi^{\frac{k}{2}} r_1^k}{k2^k \Gamma(\frac{k}{2}+1)} \tag{2.1}$$

In dimension 2 (*cf.* Figure 2.4 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.* Figure



pyramid (s,A,B,C,D)

s(0,0,0)
A(1,0,0)
B(1,0,1)
C(1,1,1)
D(1,1,0)

$\cos(AsC) = \frac{1}{\sqrt{3}} \Rightarrow angle(AsC) \approx 54,74^0$

$\cos(BsD) = \frac{1}{2} \Rightarrow angle(BsD) = 60^0$

$angle(BsD) > angle(AsC)$

Figure 2.5: Maximum angle at the apex of a pyramid.

2.4 b), each face of the cube is divided into 4 equal squares, which yields 24 cells, each supporting one pyramid (*cf.* Figure 2.4 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 non negative. Let $\mathcal{P}$ be the pyramid associated to this cell (refer to Figure 2.5 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 $\mathcal{P} \setminus \{s\}$. We wish to compute the maximum of angle $\widehat{M_1 s M_2}$. 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 α_i's and β_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 $\widehat{M_1 s M_2}$ corresponds to the minimum of:

$$\cos \widehat{M_1 s M_2} = \frac{\sum_{i=1}^{k} \alpha_i \beta_i}{\sqrt{\sum\limits_{i=1}^{k} \alpha_i^2}\sqrt{\sum\limits_{i=1}^{k} \beta_i^2}} = \frac{u}{\sqrt{u+v}\sqrt{u+w}} \tag{2.2}$$

   The numerator, u, may take integral values in the range $[1, k]$. Now, suppose $u \geq 1$ 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 $v = \lfloor \frac{k-u}{2} \rfloor$.

   Finally, since $u \rightsquigarrow \frac{u}{\sqrt{u+\lfloor \frac{k-u}{2} \rfloor}\sqrt{k-\lfloor \frac{k-u}{2} \rfloor}}$ is an increasing function on $[1, k]$ that reaches its minimum for $u = 1$, the minimum of (2.2) is:

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

   After unifying cases k odd and even, we find:

$$\forall M_1, M_2 \in \mathcal{P} \setminus \{s\},\ \cos \widehat{M_1 s M_2} \geq \frac{1}{\lfloor \frac{k+1}{2} \rfloor^{\frac{1}{2}} \lceil \frac{k+1}{2} \rceil^{\frac{1}{2}}} \tag{2.3}$$

This result may be checked in the plane and in 3-space, in Figures 2.4 and 2.5, respectively. □

Notation convention: in order to simplify the expression of subsequent results, let ζ_k represent expression $\left(\lfloor \frac{k+1}{2} \rfloor \lceil \frac{k+1}{2} \rceil\right)^{1/2}$.

Corollary 2.0.3 *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:*

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

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

Proof In the setting of Corollary 2.0.2, 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 2.2. 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 $\mathcal{P}_j$:

$$M \in \mathcal{P}_j \Rightarrow \cos \widehat{qsM} \geq \frac{1}{\zeta_k} = \frac{r_1}{t} \Rightarrow M \in \mathcal{C}_{\mathcal{B}_1}(sp, \arccos \frac{r_1}{t})$$

This implies that:

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

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

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

□

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 2.3 *Assuming the probability density f to be quasi-uniform with bounds c_1 and c_2, consider rectangle $\mathcal{R}_k \subseteq \mathcal{U}_k$ in $DT(S \cap \mathcal{R}_k) <_\Delta DT(S)$. If $s \in S \cap \mathcal{R}_k$ is a site at minimal distance t from the boundary of $\mathcal{R}_k$, then the probability $p(\mathcal{C}_1)$ for s to be unfinished in $DT(S \cap \mathcal{R}_k)$ wrt $DT(S)$ satisfies:*

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

Proof Consider ball $\mathcal{B}_1$, with center s and radius t/ζ_k, paved by pyramids $\mathcal{P}_j$, $j \in [1, k \times 2^k]$. Using Corollary 2.0.3, condition $\mathcal{C}_1$ – that s is unfinished in $DT(S \cap \mathcal{U}_k)$ *wrt* $DT(S)$ – implies condition $\mathcal{C}_2$ – that at least one of the open partitioning k-cells $\mathcal{P}_j$ is void of sites. Hence,

$$\begin{aligned}
p(\mathcal{C}_1) &\leq p(\mathcal{C}_2) \\
&\leq \sum_{j=1}^{k2^k} p(S \cap \mathcal{P}_j = \emptyset) = \sum_{j=1}^{k2^k} \Big(1 - \int_{\mathcal{P}_j} f\Big)^{N-1} \\
&\leq k2^k \times \left[1 - \frac{c_1 \pi^{\frac{k}{2}} t^k}{k2^k \zeta_k^k \Gamma(\frac{k}{2}+1)}\right]^{N-1}
\end{aligned}$$

using property $f(x_1, x_2, \ldots, x_k) \geq c_1$. □

2.5 Expected number of unfinished sites

By integrating the previous probability over a rectangular domain, we obtain an upper bound on the expected number of unfinished sites in a hyperrectangle. Lemma 2.4 and Corollary 2.0.4 provide, respectively, the computation of an integral and an upper bound on it, that are both necessary for the probabilistic evaluation in Theorem 2.1.

Lemma 2.4 *For integers $n \geq 0$ and $k \geq 2$:*

$$\mathcal{I}_n^k = \int_0^1 \left(1 - x^k\right)^n \mathrm{d}x = \prod_{i=1}^{n} \frac{ki}{ki+1} \tag{2.4}$$

Proof It is possible, through the change of variable $u = x^k$, to express $\mathcal{I}_n^k$ in terms of the **Beta** function:

$$\mathcal{I}_n^k = \frac{1}{k}\int_0^1 u^{\frac{1}{k}-1}(1-u)^{(n+1)-1}\mathrm{d}u = \frac{1}{k}\mathcal{B}(\frac{1}{k}, n+1) = \frac{\Gamma(\frac{1}{k})\Gamma(n+1)}{k\Gamma(\frac{1}{k}+n+1)}$$

$$\mathcal{I}_n^k = \frac{\Gamma(\frac{1}{k})n!}{k\left[\prod_{i=0}^{n}(\frac{1}{k}+i)\right]\Gamma(\frac{1}{k})} = \frac{n!}{\prod_{i=1}^{n}(\frac{1}{k}+i)} = \frac{\prod_{i=1}^{n} i}{\prod_{i=1}^{n}\frac{ki+1}{k}} = \prod_{i=1}^{n}\frac{ki}{ki+1}$$

□

Corollary 2.0.4 *For integers $n \geq 1$ and $k \geq 2$:*

$$\mathcal{I}_n^k = \int_0^1 \left(1 - x^k\right)^n \mathrm{d}x < \frac{1}{\sqrt[k]{n+1}} \tag{2.5}$$

Proof Let us show this result by induction on n, for fixed k.
For $n = 1$:

$$\mathcal{I}_1^k < \frac{1}{\sqrt[k]{2}} \Leftrightarrow \frac{k}{k+1} < \frac{1}{\sqrt[k]{2}} \Leftrightarrow (k+1)^k > 2 \times k^k$$

$$\Leftrightarrow k^k + kk^{k-1} + C_k^2 k^{k-2} + \cdots > k^k + k^k$$

Assuming the property true up to $(n-1)$, let us show it is also true for n:

$$\mathcal{I}_n^k < \frac{1}{\sqrt[k]{n+1}} \Leftrightarrow \frac{kn}{kn+1}\mathcal{I}_{n-1}^k < \frac{1}{\sqrt[k]{n+1}}$$

$$\Leftrightarrow \mathcal{I}_{n-1}^k < \frac{kn+1}{kn}\frac{1}{\sqrt[k]{n+1}}$$

By induction hypothesis, $\mathcal{I}_{n-1}^k < \frac{1}{\sqrt[k]{n}}$, whence:

$$\frac{1}{\sqrt[k]{n}} < \frac{kn+1}{kn}\frac{1}{\sqrt[k]{n+1}} \Leftrightarrow \left(\frac{kn+1}{kn}\right)^k > \frac{n+1}{n}$$

$$\Leftrightarrow (kn+1)^k > k^k n^k + k^k n^{k-1}$$

$$\Leftrightarrow k^k n^k + kk^{k-1}n^{k-1} + C_k^2 k^{k-2}n^{k-2} + \cdots > k^k n^k + k^k n^{k-1} \qquad \square$$

Remark: [Luc Devroye] The convexity of $x \rightsquigarrow \log\Gamma(x)$, yields

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

Hence, $\mathcal{I}_n^k$ differs from the upper bound we provide ($(n+1)^{-1/k}$) by a factor of at most $\frac{1}{\Gamma(1+1/k)}$ (the bound is reached for $k = 1$ and $k \to +\infty$). For k in the range $[1, +\infty]$, $1 \leq \frac{1}{\Gamma(1+1/k)} < 1,15$, and the error never exceeds 15%.

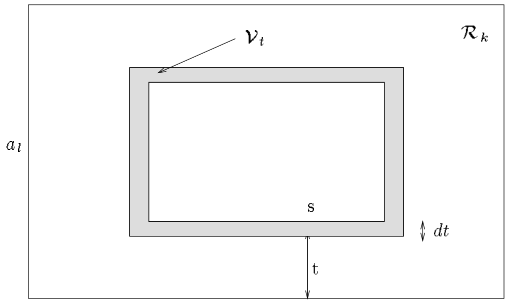


Figure 2.6: Expected number of unfinished sites in a k-rectangle.

Theorem 2.1 *In the setting of Lemma 2.3, 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:*

$$E(\mathcal{R}_k) \leq \frac{N}{\sqrt[k]{N}} \frac{c_2}{\sqrt[k]{c_1}} \mathcal{S}_k \frac{k2^{k+1}\zeta_k \left(k\Gamma(\frac{k}{2}+1)\right)^{1/k}}{\sqrt{\pi}} \tag{2.6}$$

Proof Let a_1, a_2, ..., 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 $0 \leq t \leq \frac{a_l}{2}$) (Figure 2.6). Let us call $\mathcal{S}_t$ the exterior surface of $\mathcal{V}_t$:

$$\mathcal{S}_t = 2\sum_{j=1}^{k}\big(\prod_{i\neq j}(a_i - 2t)\big) \leq \mathcal{S}_k = 2\sum_{j=1}^{k}\prod_{i\neq j} a_i$$

which implies:

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

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

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

Using Lemma 2.3, we have the following estimation:

$$\begin{aligned} E(\mathcal{R}_k) &\leq N\int_0^{\frac{a_l}{2}} k2^k\Big[1 - \frac{c_1\pi^{\frac{k}{2}}t^k}{k2^k\zeta_k^k\Gamma(\frac{k}{2}+1)}\Big]^{N-1} c_2\mathcal{S}_k\mathrm{d}t \\ &= N\frac{c_2}{\sqrt[k]{c_1}}\mathcal{S}_k\,\frac{k2^{k+1}\zeta_k(k\Gamma(\frac{k}{2}+1))^{1/k}}{\sqrt{\pi}} \times \int_0^{\frac{a_l\sqrt{\frac{\pi}{4}}\sqrt[k]{c_1}}{\zeta_k(k\Gamma(\frac{k}{2}+1))^{1/k}}} \left(1-x^k\right)^{N-1}\mathrm{d}x \end{aligned}$$

with the change of variable:

$$x = t\Big[\frac{c_1\pi^{k/2}}{k2^k\zeta_k^k\Gamma(\frac{k}{2}+1)}\Big]^{1/k}.$$

Since $a_l \leq 1$, $c_1 \leq 1$ and $k \geq 2$, we may write:

$$\begin{aligned} E(\mathcal{R}_k) &\leq N\frac{c_2}{\sqrt[k]{c_1}}\mathcal{S}_k\,\frac{k2^{k+1}\zeta_k\Big[k\Gamma(\frac{k}{2}+1)\Big]^{1/k}}{\sqrt{\pi}} \times \int_0^1 (1-x^k)^{N-1}\mathrm{d}x \\ &\leq \frac{N}{\sqrt[k]{N}}\frac{c_2}{\sqrt[k]{c_1}}\mathcal{S}_k\,\frac{k2^{k+1}\zeta_k\Big[k\Gamma(\frac{k}{2}+1)\Big]^{1/k}}{\sqrt{\pi}} \end{aligned}$$

using the upper bound on the integral from Lemma 2.4. Hence,

$$E(\mathcal{R}_k) = O\Big(\frac{N}{\sqrt[k]{N}}\frac{c_2}{\sqrt[k]{c_1}}\mathcal{S}_k\Big) = O(N^{1-1/k})$$

□

Remark:

1. This bound improves upon that obtained by Katajainen and Koppinen in the plane [18].
2. Luc Devroye also pointed out to us that Theorem 2.1 should generalize to any convex set. In this case, $\mathcal{S}_k$ would be the surface of the set.

3

Theoretical results

3.1 Average-case analysis of a class of k-dimensional merge procedures using the notion of unfinished sites

In this section, we first show how the previous result may be exploited to analyze certain classes of multi-dimensional *divide-and-conquer*-type algorithms. Then we present an application to the construction of the Euclidian Delaunay triangulation in dimension 2, that performs extremely well in practice.

Please refer to Figures 3.1 and 3.2.

Theorem 3.1 *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 ($c_1 \leq c_2$). 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 ([4]);*

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 (wrt $\mathcal{U}_k$), then the whole merge *phase of the algorithm will be proportional to N.*

Proof Using Theorem 2.1, the running time of the *merge* phase will be proportional to

$$\frac{N}{\sqrt[k]{N}} \frac{c_2}{\sqrt[k]{c_1}} \mathcal{S}_k \frac{k 2^{k+1} \sqrt{\lceil \frac{k+1}{2} \rceil \lfloor \frac{k+1}{2} \rfloor} \sqrt[k]{k \Gamma(\frac{k}{2}+1)}}{\sqrt{\pi}} \sum_{k-rect.} \mathcal{S} \qquad (3.1)$$

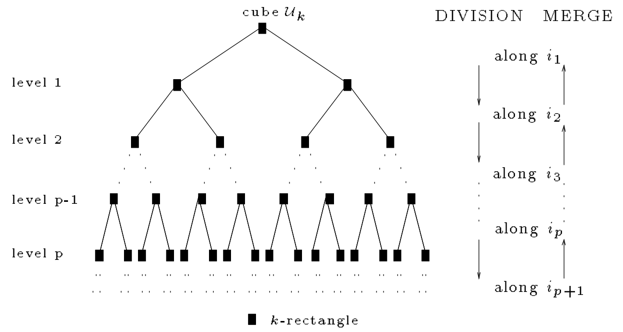


Figure 3.1: Tree representation of divides and conquers according to a balanced k-d tree.

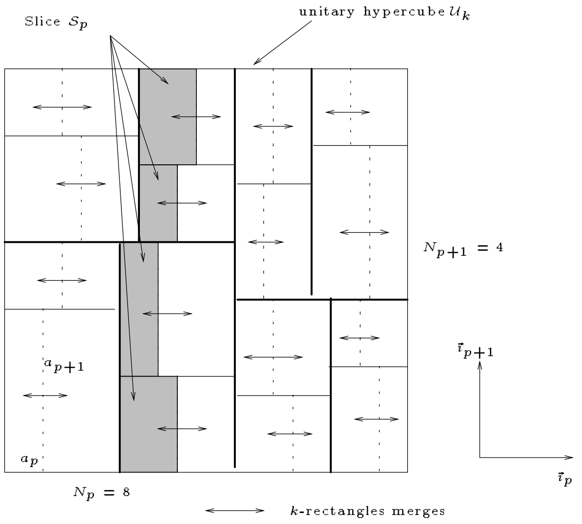


Figure 3.2: Merges at level p: slice S_p with all its (shaded) hyperrectangle constituents.

Hence, we must evaluate $\sigma = \sum_{k-rect.} \mathcal{S}$, this quantity representing the sum of the surfaces of all k-rectangles involved in the merges. We shall decompose σ into sub-sums σ_p, one for each merge level p (Figure 3.1).

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 3.1 *Let us define* slice $\mathcal{S}_j$ *relative to direction* $\vec{\imath}_j$ *as one subset of the hyperrectangles partitioning* $\mathcal{U}_k$*, such that the orthogonal projection of* $\mathcal{S}_j$ *on either hyperface of* $\mathcal{U}_k$ *orthogonal to* $\vec{\imath}_j$ *constitutes a partition of this hyperface (Figure 3.2).*

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_j the number of slices relative to direction $\vec{\imath}_j$.

Consider merge-level p. Save for the last level – for which this value is only an upper bound for N_j – a simple induction argument yields:

$$N_j = 2^{\lceil \frac{p-j+1}{k} \rceil}.$$

Let us now evaluate σ_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 $2\sum_{j=1}^{k} N_j$, whence:

$$\frac{1}{2}\sigma_p \leq \sum_{j=1}^{k} 2^{\lceil \frac{p-j+1}{k} \rceil} \leq k2^{\lceil \frac{p}{k} \rceil}$$

Summing over all h merge-levels yields:

$$\frac{1}{2}\sigma = \frac{1}{2}\sum_{p=1}^{h} \sigma_p \leq \sum_{p=1}^{h} k2^{\lceil \frac{p}{k} \rceil} = k\sum_{p=1}^{h} 2^{\lceil \frac{p}{k} \rceil}$$

$$\leq k(k\sum_{w=1}^{\lceil \frac{h}{k} \rceil} 2^w) \leq k^2 2^{\lceil \frac{h}{k} \rceil + 1} \leq 4k^2 2^{\lceil \frac{h}{k} \rceil - 1}$$

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

$$\sqrt[k]{N} > 2^{\frac{h-1}{k}} \geq 2^{\lceil \frac{h}{k} \rceil - 1} \Rightarrow \sigma = \sum_{k-rect.} \mathcal{S} \leq 8k^2 \sqrt[k]{N}$$

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:

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

□

3.2 Application: 2D Delaunay triangulation in linear expected time after two-directional sorting

3.3 Reasons for hope

In the standard algorithm (Lee and Schachter [19]), the recursive process splits the original point set into strips of decreasing "widths", until reaching elementary "cells" containing only one site. This stage is naturally followed by the successive merging of pairs of adjacent sub-triangulations, in reverse order of the division process.

However, in the overall merge process, a considerable effort is put into creating many triangles (pertaining to long and narrow adjacent strips), that stand no chance to be part of the actual completed Delaunay triangulation, due to their exceedingly distorted shapes (see Figure 4.2), and that must, later on, be destroyed and replaced with "better-looking" candidates for final qualification. Ohya, Iri and Murota ([32]) have shown that *even* the average-case running time for the entire standard merge step is $\Omega(N \log N)$, an inherent consequence of the strip-wise division principle.

3.4 Optimized algorithm

We shall use the term "optimized algorithm" [22, 21] to refer to the algorithm based on a balanced 2-d tree, as derived from Section 3.1. The main difference between the optimized and standard (Lee & Schachter's) algorithms lies in the way the recursive process divides the original point set.

3.4.1 Two-directional division

Let M_x and M_y be the abscissa and the ordinate of point M. Define relations $<_X$ and $<_Y$ by:

$$M <_X N \Leftrightarrow ((M_x < N_x) \text{ or } (M_x = N_x \text{ and } M_y < N_y))$$

$$M <_Y N \Leftrightarrow ((M_y < N_y) \text{ or } (M_y = N_y \text{ and } M_x > N_x))$$

Let A and B be two planar point sets. We shall say that $A <_X B$ (resp. $A <_Y B$) if and only

$$\forall a \in A,\ \forall b \in B,\ a <_X \ (\text{resp. } <_Y)\ b.$$

We shall also say that A and B are vertically (resp. horizontally) separable, in that case.

A planar point set A is said to be X-divided (resp. Y-divided) into A_1 and A_2 if and only if $A_1 <_X A_2$ (resp. $A_1 <_Y A_2$), with $A_1 \cup A_2 = A$ and if A_1 and A_2 have the same number of points (up to one), as shown on Figure 3.3. Note that A_1 or A_2 may be empty.

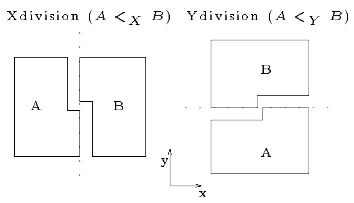


Figure 3.3: X- and Y- divisions.

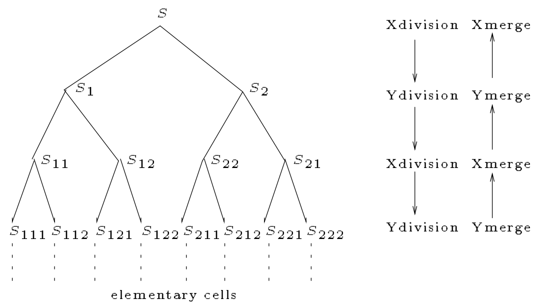


Figure 3.4: Tree representation of divisions and merges.

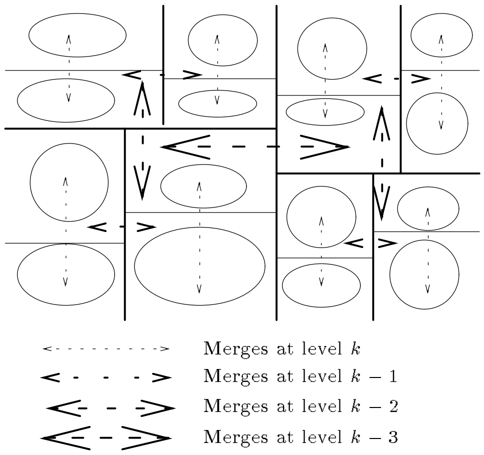


Figure 3.5: Merging process.

We now wish to divide the sites according to the dynamics of a *balanced 2-d tree* ([4]; see also [17, 20] for other developments on such structures). At the root of the 2-d tree, the point set S is X-divided into S_1 and S_2 (refer to Figure 3.4). At the next level, S_1 and S_2 are Y-divided into S_{11}, S_{12} and S_{21}, S_{22}, respectively. The process is continued, alternating X-divisions and Y-divisions, until reaching elementary cells containing one single site.

The 2-d tree "mechanism" is used to support the Delaunay divide phase, but no such structure needs actually be constructed by the program, although one may wish to use an array to store the order information for efficient subsequent median partitioning.

In all cases, the overall cost for setting up this division scheme is $\Theta(N \log N)$, whatever optimal median-finding-based implementation is chosen. As the reader will check in the result section, this step is considerably less time-consuming than the actual Delaunay construction: it takes much less *machine* time to iterate the 2-d tree construction loop than to iterate the Delaunay-merge loop!

3.4.2 Two-directional Delaunay merge

Next, elementary cells (and then strips) are merged in reverse order of their creation. Subsets obtained through X-division (Y-division) are merged through X-merge (Y-merge), (*cf.* Figure 3.5).

This is possible mainly because Lee and Schachter's merging technique is very general and

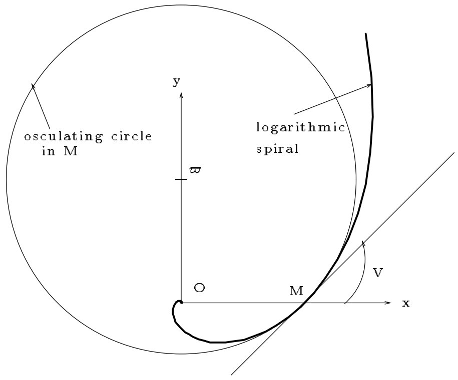


Figure 3.6: Properties of the logarithmic spiral.

operates independently of the dividing line's direction: generalizing H. Edelsbrunner's remark ([13, p. 146]), all we need, to merge two linearly separable (convex) point set triangulations, is (one of) the site(s) closest to the dividing line, be it horizontal or vertical. Such sites may be retrieved in constant-time if we store pointers to the topmost, bottommost, leftmost and rightmost (according to the X- and Y-order relations) convex hull edges in each triangulation. Practical considerations for implementing this two-directional merge function may be found in the first author's Ph.D. Thesis ([20]).

3.4.3 Overview of the algorithm

Step 1 : 2-D TREE CONSTRUCTION
Step 2 : TRIANGULATION
  DELAUNAY (S)
    XDIVIDE(S);
  XDIVIDE(S)
    If ($|S| > 1$)
      XPARTITION (S, S_1, S_2);
      YDIVIDE (S_1);
      YDIVIDE (S_2);
      XMERGE(S_1, S_2);
  YDIVIDE(S)
    If ($|S| > 1$)
      YPARTITION (S, S_1, S_2);
      XDIVIDE (S_1);
      XDIVIDE (S_2);
      YMERGE(S_1, S_2);

3.5 Computational complexity

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

3.5.1 Worst-case complexity

As emphasized in 3.4.1, we already know that the preprocessing two-directional sorting phase may be done in $\Theta(N \log N)$. We also know ([16]) 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

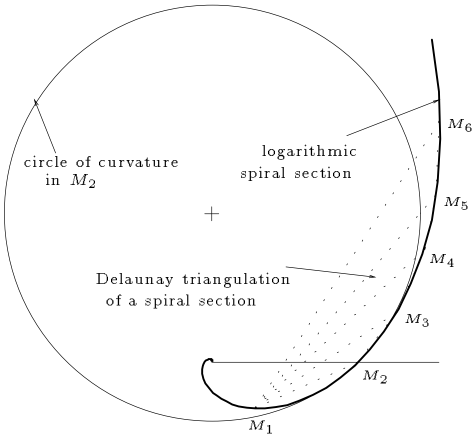


Figure 3.7: Triangulation of a logarithmic spiral section: inserting a new site may cause the destruction of all already constructed Delaunay triangles...

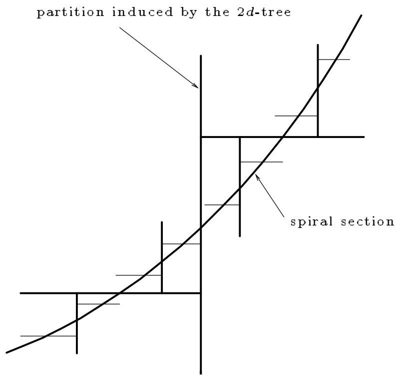


Figure 3.8: Spiral partitioning induced by a 2-d tree: merging two adjacent cells may cause the destruction of all triangles in either...

show that the worst-case running time of Step 2 is $\Omega(N \log N)$. For this purpose, we shall study the Delaunay triangulation on N sites distributed over a section of a logarithmic spiral.

Consider the logarithmic spiral Γ_a with equation $\rho = ae^{m\theta}$ in polar coordinates, and note M_i the point of the curve associated with θ_i. This spiral has many properties, among which (Figure 3.6):

1. The tangent in M makes a constant angle V with line OM ($tanV = \frac{1}{m}$).
2. Let ϖ be the center of curvature at point M. The circle through M and centered in ϖ contains the entire section of Γ *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 Γ *before* M_k; the entire section of Γ *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 $M_i, 1 \leq i \leq N$ on a section of spiral, consists in linking M_1 to all other sites M_j ($j \neq 1$), then linking each site $M_i, 1 \leq i < N$ 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 (Figure 3.7).

Consider, for example, the logarithmic spiral Γ_1 with equation $\rho = e^\theta$.

$$\begin{cases} x(\theta) = e^\theta \cos\theta \\ y(\theta) = e^\theta \sin\theta \end{cases} \Rightarrow \begin{cases} x\prime(\theta) = \sqrt{2}e^\theta \cos(\theta + \frac{\pi}{4}) \\ y\prime(\theta) = \sqrt{2}e^\theta \sin(\theta + \frac{\pi}{4}) \end{cases}$$

Consider the set $\{M_i\}_{i \in [1,N]}$ of sites with $-\frac{\pi}{4} \leq \theta_1 < \ldots < \theta_N \leq \frac{\pi}{4}$, on Γ_1. Clearly, if $1 \leq i < j \leq N$,

$$M_{i_x} < M_{j_x}, \text{ and } M_{i_y} < M_{j_y}.$$

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 $<_X$ and $<_Y$), or else *after* (Figure 3.8). 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:

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

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.

3.5.2 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 [10] (which is possible thanks to the quasi-uniform distribution assumption), the construction of the 2-d tree remains in $\Omega(N \log N)$.

However, the expected running time of Step 2 (page 21) is improved. That the arguments of Section 3.1 hold for this algorithm is justified by the following theorem:

Theorem 3.2 *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.*

proven by R. Dwyer ([12]), using a previous result by Guibas and Stolfi ([16]).

This theorem implies that, in the case of a quasi-uniform distribution, the more general result of Theorem 3.1 yields an $O(N)$ expected running time for Step 2, in a straighforward manner (the interested reader will find a specific analysis for the $2D$ case in [21]).

4

Experimental results and comparison with other algorithms

4.1 Experimental results

In general, the optimized algorithm proves to be 2 to 3 times faster than the standard one. For N distincts sites, the total number of edges created and of edges destroyed is close to $4N$ and N, respectively, and the number of edges of the initial triangulation still valid in the final one is close to $3N$.

Extensive tests were conducted on sets of up to seven millions sites, uniformly distributed in a square domain. Both the standard algorithm, and the standard-based optimized algorithm 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.

Figure 4.1 illustrates the results obtained on the C3. These results strongly corroborate the theory, and show the asymptotic $O(N \log N)$ and $O(N)$ behaviour of the standard and optimized versions, respectively (excluding preprocessing sorts).

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 also frees interpretations from considerations on the speed of the processors, loading factors, and so forth. The difference between the running times of both versions becomes perceptible from 60 sites on, and, for 130,000 sites, the optimized version is already twice as fast as the standard one. 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. The rate of triangulation (exclusive of sorting) is about 30,000 sites per second (between 50,000 and 60,000 triangles).

4.2 Shewchuk's implementation

In [29], J.R. Shewchuk describes an idea similar to the one developed in Section 3.2 above, 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, [28, 27], 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 ([19]). As a matter of fact, as the latter merges very thin and narrow strips, it must often determine the position of a site

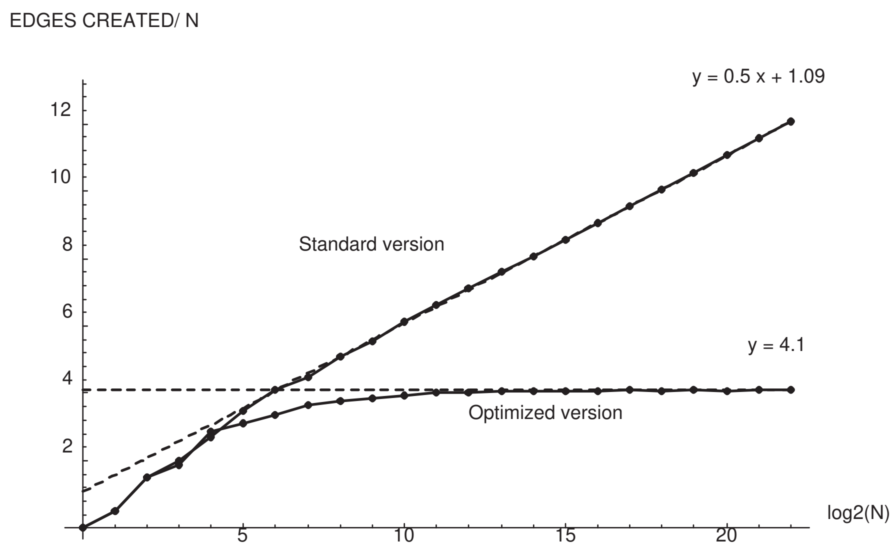


Figure 4.1: Comparison between the standard version (Lee and Schachter), and our optimized version.

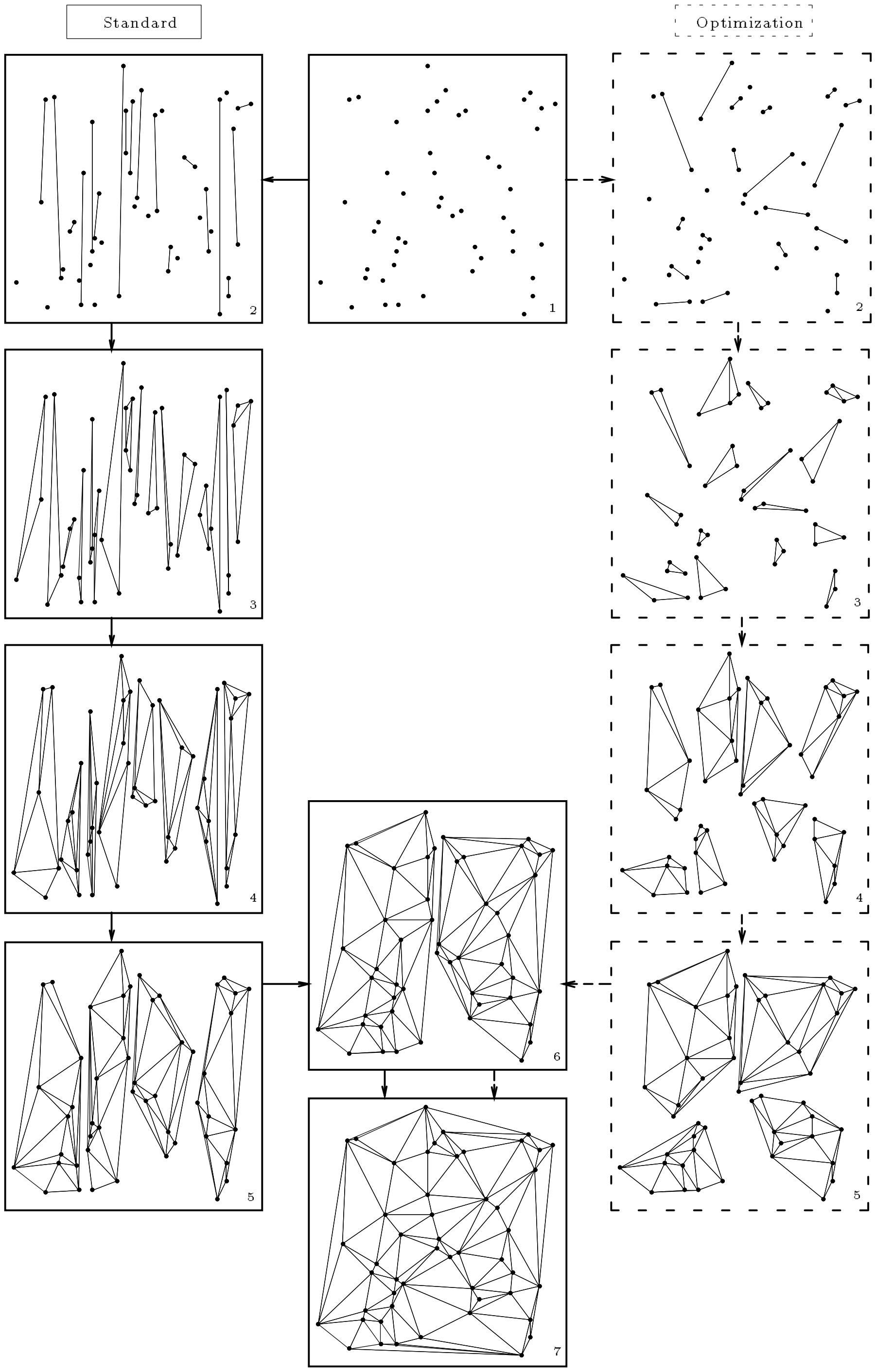


Figure 4.2: Successive stages for the standard (solid arrows) and optimized (dotted arrows) versions for the Delaunay triangulation of 50 sites. On the left-hand side, the triangles are thinner and stand almost no chance to be Delaunay triangles. On the right-hand side, the triangles are fatter.

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 Figure 4.2).

4.3 Comparison with other algorithms

According to Su and Drysdale [30, 31], the reputedly best algorithms are those of Katajainen and Koppinen (**KK**) [18], and Dwyer (**Dw**) [12], even for non uniform distributions. We confirm this fact for *uniform distributions* only (see the top diagrams of Figures 4.3 and 4.4). Our algorithm (**2d**), based on a 2-d tree, is only slightly slower. The two other algorithms use hashing techniques, and it is well known that hashing performs best on uniform data and is faster than sorting. The algorithm (**a2d**) [3] is based on an adaptive 2-d tree [15], 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. Lee and Schachter's algorithm (**LS**) [19] is slower than the latter: its triangulation step is in $\Theta(N \log N)$!

Now, for a *non-uniform distribution* with clusters (normal distribution $N(0,\ 0.01)$ at 10 randomly chosen sites in the unit square), the algorithm based on a 2-d tree performs better than Dwyer's algorithm, and Katajainen and Koppinen's algorithm whose behaviour resembles Lee and Schachter's algorithm. Our algorithm is more appropriate to these situations (see the bottom diagrams of Figures 4.3 and 4.4).

The interested reader will find other comparisons with other algorithms, and for other distribution types, in C. Lemaire's Ph.D. Thesis ([20]).

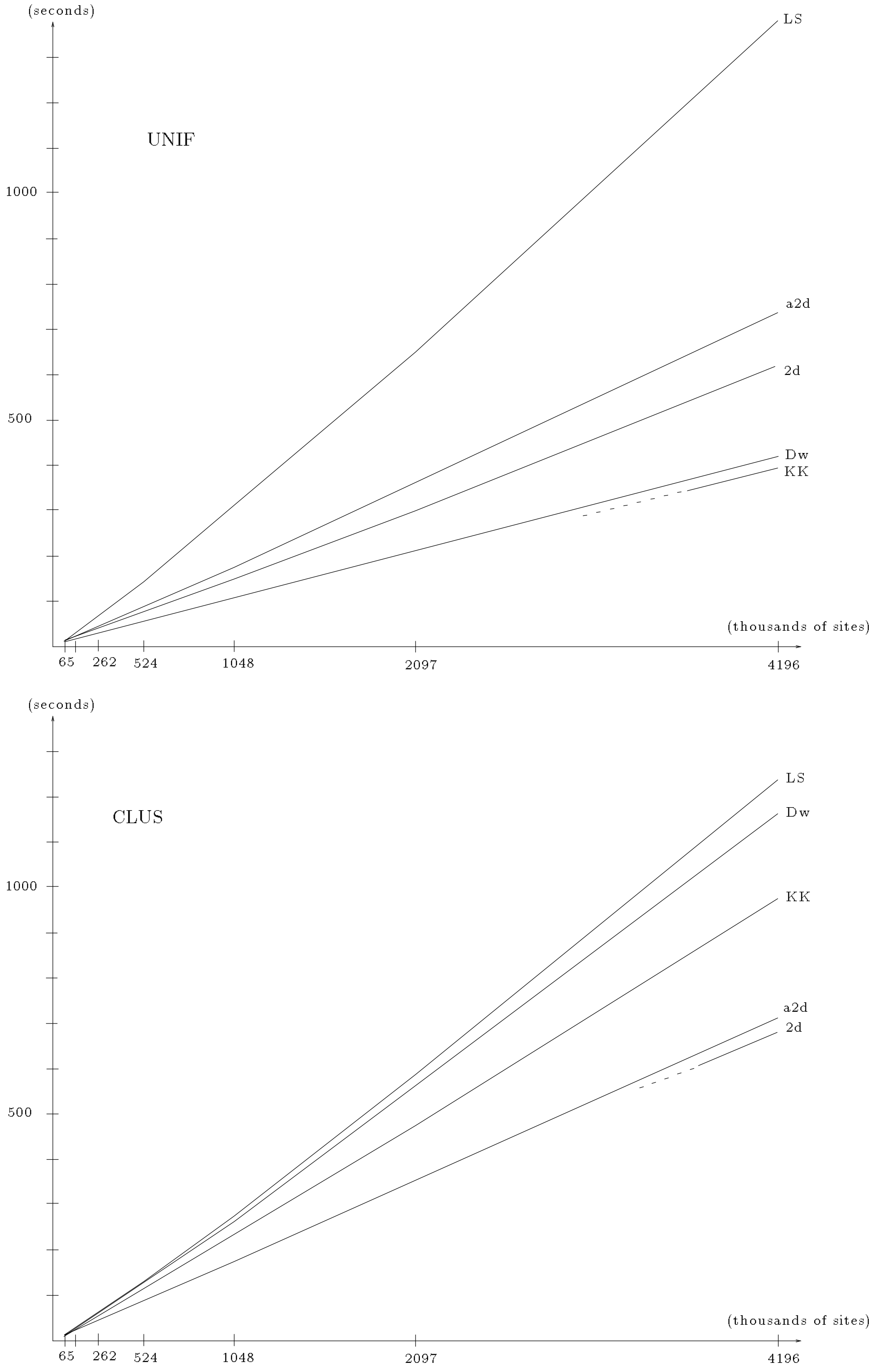


Figure 4.3: Comparison of the running time of Delaunay triangulation algorithms: the algorithms are tested on a *Convex* C3 – a super-computer with 2 Gb of memory – on data sets of up to seven millions sites distributed uniformly (top diagram) or non-uniformly with clusters (bottom diagram) in a square domain.

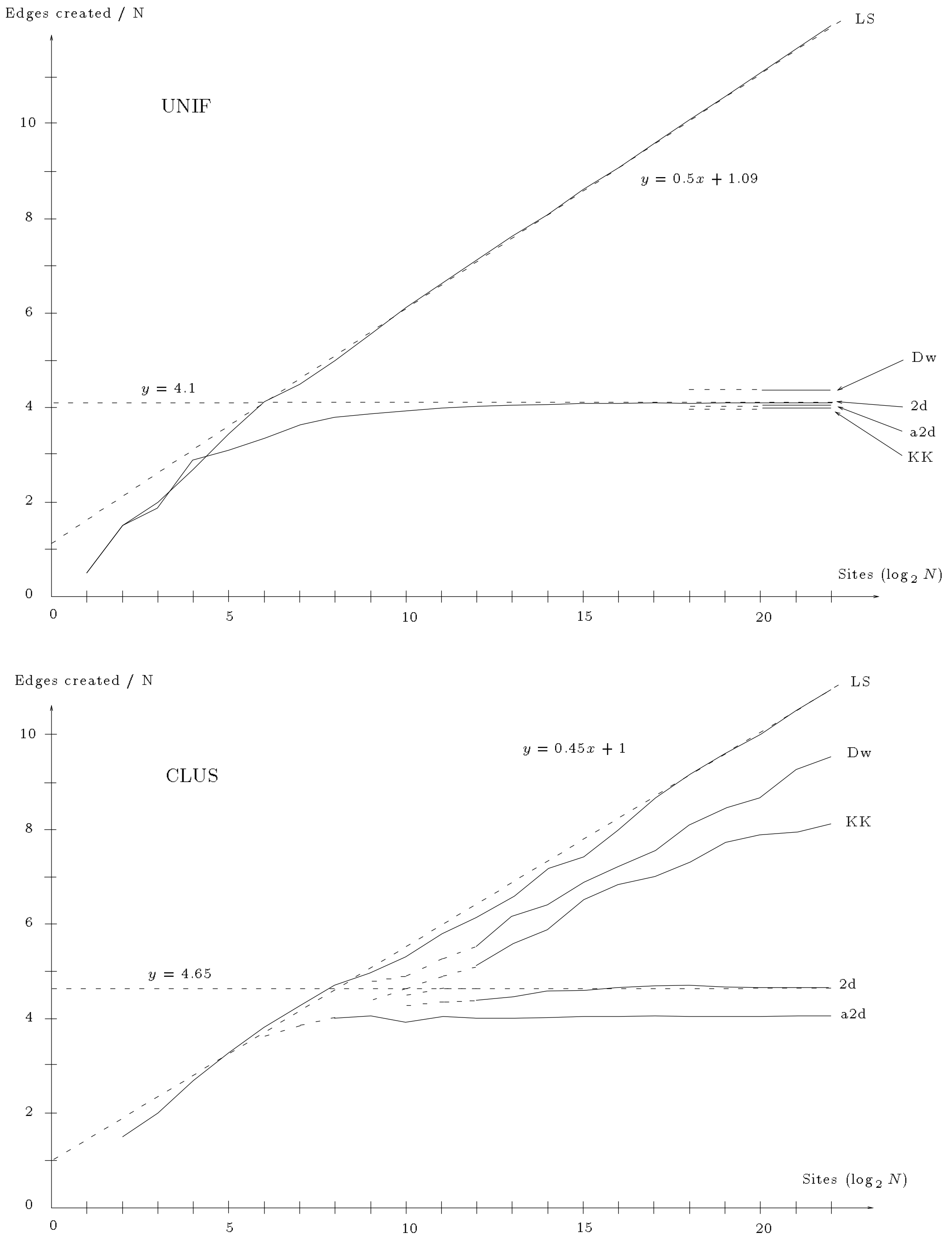


Figure 4.4: Comparison of Delaunay triangulation algorithms according the total number of edges created, in the same conditions as for the previous figure.

5

Conclusion and open problems

5.1 Delaunay triangulation in any dimension

We have just seen that the idea presented in this paper could be applied to the two-dimensional case with an effective speed-up of the standard 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, Montani et Scopigno [24] 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. research of M. Elbaz [14] and B. Adam [1] present an algorithm for constructing Delaunay triangulations in 3-space that is truly of the *divide-and-conquer* type. The process of merging sub-triangulations is 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. C. E. Buckley [8] 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 entreprise.

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.

On the basis of our results, the chances are great that such an algorithm would be very efficient, and might compete with Dwyer's algorithm ([11]), which uses a pre-partitioning of the sites *via* a regular grid. However, we are fully aware that merging sub-triangulations in higher dimensions is a tantalizing task! In $3D$, the algorithm is almost established [1].

5.2 Perspectives

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, Eppstein and Yao ([7]) on the Delaunay triangulation in any dimension.

One of our next research goals is to prove Theorem 3.1 for other classical distributions. Another goal is to prove that the method may be applied in k-dimensional setting, to obtain, after multi-dimensional pre-sorting, a triangulation algorithm, the expected running time of which would be proportional to the number of sites. This paper has shown the efficiency of such a scheme in the planar case. The corresponding algorithm is fairly easy to implement, and seems to be one of the fastest, even for non-uniform distributions.

An important question is to know whether, in dimension k, sorting the sites along k directions does not lower the upper bound on the construction of their Euclidian Delaunay triangulation. This problem was mentioned for dimension 2 by O. Devillers (*INRIA*, Sophia Antipolis), a few years ago. This remains an open problem, whatever the dimension of the space (except the trivial case of dimension 1, where the Delaunay triangulation consists in linking the sites in sorted order). Unfortunately, the results in this paper do not allow to lower this upper bound, even in the plane.

Seidel [25] has shown that after sorting by x-coordinates, computing a $2d$ Delaunay triangulation still requires $\Omega(N \log N)$ time. However, if one uses the Minkowski distance associated to a convex polygon (triangle in particular, distance $L_1, \ldots$), Chew et Fortune ([9]) have shown that computing the Delaunay triangulation in the plane may be done in time $O(N \log\log N)$ after sorting the sites by abscissæ and ordinates.

Finally, in [12], Dwyer proposed a $2d$ Euclidian Delaunay triangulation algorithm with $O(N \operatorname{loglog} N)$ 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 & Schachter's algorithm ([19]).
2. Triangulate each row by merging pairs of adjacent cells.
3. Merge lines 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, on average. Using Theorem 2.1, 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 line). There is a strong experimental evidence that the triangulation phase of lines (containing up to 8 million sites) is linear. The rigorous proof of such a conjecture remains an open problem.

Acknowledgments

The authors wish to thank Olivier Devillers (*INRIA*, Sophia Antipolis, France) for his useful comments on an early version of this paper, and Luc Devroye (McGill University, Montreal, Canada) for his help, encouragements, and many comments, among which Remarks 2.5 and 2.5.2.

Bibliography

[1] B. Adam. *Construction des diagrammes de Delaunay dans le plan et dans l'espace.* PhD thesis, Université de Haute-Alsace, France, no 453, 1996. 208 pages.

[2] F. Aurenhammer. Voronoi diagrams – A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, September 1991.

[3] M. Elbaz B. Adam and J.C. Spehner. Construction du diagramme de Delaunay dans le plan en utilisant les mélanges de tris. Quatrièmes Journées de l'AFIG, Dijon (France).

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[5] M. Berger. *Géométrie III. Convexes et polytopes, polyèdres réguliers, aires et volumes.* Fernand Nathan, Paris, 1974.

[6] M. Berger. *Geometry.* Springer-Verlag, Berlin, 1987.

[7] M. Bern, D. Eppstein, and F. Yao. The expected extremes in a Delaunay triangulation. *Internat. J. Comput. Geom. Appl.*, 1:79–91, 1991.

[8] C. E. Buckley. A divide-and-conquer algorithm for computing 4-dimensional convex hulls. In *Proc. Computational Geometry and its Applications*, volume 333 of *Lecture Notes Comput. Sci.*, pages 113–135. Springer-Verlag, 1988.

[9] L. P. Chew and S. Fortune. Sorting helps for Voronoi diagrams. In *Proc. 13th Symposium on Mathematical Programming, Japan*, 1988. Also: CS-TR 93-1347, Department of Computer Science, Cornell University, Ithaca, May 1993, 13 pages, and *Algorithmica*, vol. 18, 1997, pages 217–228.

[10] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* M.I.T. Press, McGraw-Hill Book Company, Cambridge, Mass., New York, N.Y., 1989.

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

[12] R.A. Dwyer. A faster divide-and-conquer algorithm for constructing Delaunay triangulations. *Algorithmica*, 2:137–151, 1987.

[13] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, Heidelberg, West Germany, 1987.

[14] M. Elbaz. *Les diagrammes de Voronoi et de Delaunay dans le plan et dans l'espace.* PhD thesis, Université de Haute-Alsace, no 209, France, 1992.

[15] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3:209–226, 1977.

[16] L.J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphic*, 4:74–123, 1985.

[17] P. Houthuys. Box sort, a multidimensional binary sorting method for rectangular boxes, used for quick range searching. *The Visual Comput.*, 3(4):236–249, December 1987.

[18] J. Katajainen and M. Koppinen. Constructing Delaunay triangulations by merging buckets in quadtree order. *Annales Societatis Mathematicae Polonae, Series IV, Fundamenta Informaticae*, 11(3):275–288, 1988.

[19] D.T. Lee and B.J. Schachter. Two algorithms for constructing a Delaunay triangulation. *International Journal of Computing Information Sciences*, 9:219–242, 1980.

[20] C. Lemaire. *Triangulation de Delaunay et arbres multidimensionnels.* PhD thesis, ENSM.SE, Saint-Étienne (France), 1997.

[21] C. Lemaire and 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):317–336, Hermès, September 1997.

[22] C. Lemaire and 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), March 1996.

[23] A. Maus. Delaunay triangulation and the convex hull of n points in expected linear time. *BIT*, 24:151–163, 1984.

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

[25] R. Seidel. A method for proving lower bounds for certain geometric problems. In G. Toussaint, editor, *Computational Geometry*, pages 319–334. North Holland, Amsterdam (Netherlands), 1985.

[26] M.I. Shamos and D. Hoey. Closest-point problems. In *Proceedings of the 16th Annual IEEE Symposium on FOCS*, pages 151–162, 1975.

[27] 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.

[28] J. R. Shewchuk. Robust adaptive floating-point geometric predicates. In *Proceedings of the 12th ACM Annual Symposium on Computational Geometry*, pages 141–150, May 1996.

[29] J. R. Shewchuk. Triangle: engineering a $2d$ quality mesh generator and Delaunay triangulator. In *First ACM Workshop on Applied Computational Geometry*, pages 124–133, May 1996.

[30] P. Su and R.L.S. Drysdale. A comparison of sequential Delaunay triangulation algorithms. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 61–70, 1995.

[31] P. Su and R.L.S. Drysdale. A comparison of sequential Delaunay triangulation algorithms. *Computational Geometry, Theory and Applications*, 7:361–386, April 1997.

[32] M. Iri T. Ohya and K. Murota. Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms. *Journal of the Operations Research Society of Japan*, 27:306–337, 1984.