Self-Adapting Point Location

Pedro M. M. de Castro, Olivier Devillers

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Self-Adapting Point Location

Pedro Machado Manhães de Castro — Olivier Devillers

N° 7132 — version 2

initial version December 2009 — revised version Mars 2010

ISSN 0249-6399 ISRN INRIA/RR--7132--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Self-Adapting Point Location

Pedro Machado Manhães de Castro , Olivier Devillers

Thème : Algorithmique, calcul certifié et cryptographie
Équipe-Projet Geometrica

Rapport de recherche n° 7132 — version 2 — initial version December 2009
— revised version Mars 2010 — 24 pages

Abstract: Point location in spatial subdivision is one of the most studied problems in computational geometry. In the case of triangulations of $\mathbb{R}^d$, we revisit the problem to exploit a possible coherence between the query-points.

For a single query, walking in the triangulation is a classical strategy with good practical behavior and expected complexity $O(n^{1/d})$ if the points are evenly distributed. For a batch of query-points, the main idea is to use previous queries to improve the current one; we compare various strategies that have an influence on the constant hidden in the big-O notation.

Still regarding the complexity of a query, we show how the Delaunay hierarchy can be used to answer, under some hypotheses, a query q with a $O(\log \#(pq))$ randomized expected complexity, where $\#(s)$ indicates the number of simplices crossed by the line segment s, and p is a previously located query. The data-structure has $O(n \log n)$ time complexity and $O(n)$ memory complexity.

Key-words: Point location, Delaunay triangulation

This work is partially supported by ANR Project *Triangles* and Région PACA.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Localisation de points s'adaptant aux requêtes

Résumé : La localisation de points dans une subdivision de l'espace est un classique de la géométrie algorithmique, nous réexaminons ce problème dans le cas des triangulations de $\mathbb{R}^d$ pour exploiter une éventuelle cohérence entre les requêtes.

Pour une requête, marcher dans la triangulation est une stratégie classique de localisation qui donne de bons résultats pratique et a une complexité moyenne $O(n^{1/d})$ si les points sont uniformément distribués. Pour des paquets de requêtes, l'idée principale est d'utiliser les requêtes précédentes pour améliorer la requête courante; nous comparons différente stratégies qui ont une influence sur les constantes cachées dans les grands O.

Toujours à propos de la complexité d'une requête, nous montrons que la hiérarchie de Delaunay peut être utilisée pour localiser un point q à partir d'une requête précédente q avec une complexité randomisée $O(\log \#(pq))$ pourvu que la triangulation vérifie certaines hypothèses ($\#(s)$ désigne le nombre de simplex traversés par le segment s). La structure de donnée a une taille $O(n)$ et un coût de construction $O(n \log n)$.

Mots-clés : Localisation, Triangulation de Delaunay

1 Introduction

Point location in spatial subdivision is one of the most classical problems in computational geometry [13]. Given a query-point q and a partition of the d-dimensional space in regions, the problem is to retrieve the region containing q. This paper addresses the special case where the spatial subdivision is a simplicial decomposition satisfying some hypotheses.

In two dimensions, locating a point has been solved in optimal $O(n)$ space and $O(\log n)$ worst-case query time a quarter of a century ago by Kirkpatrick's hierarchy [16]. In some applications however, query-points are contained within a region, which is smaller than the whole domain in which the triangulation is defined; thus, optimality of the worst case does not necessarily translate into good performances. For example, in geometric information system the data base contains some huge geographic area, while the user usually is exploring some small region of interest; in that case queries are spatially coherent. Another example is the Poisson surface reconstruction method [1], which uses point dichotomy to find the solution to some equation; here also, the queries have some spatial coherence. For such applications, we can imagine a point location algorithm which is capable to adapt to the query distribution. A point location data-structure that benefits from the coherence of the queries is called a *self-adapting* data-structure.

Self-adapting data-structures have been successfully formalized either in terms of the *entropy* of the query distribution or by some special distances between two queries. In two dimensions, we mention: Arya *et al.* data-structure [2] or Iacono data-structure [14], both achieving a query time proportional to the entropy of the distribution of the queries, and linear space; or Iacono and Langerman's data-structure [15] and Demaine *et al.* data-structure [7], where the location time is logarithmic in terms of the distance between two successive queries for some special region-counting distances.

Despite the good theoretical behavior of the above-mentioned methods, alternative methods using simpler data-structures are still used by practitioners. Amongst these methods, *walking* from a simplex to another using the neighborhood relationships between simplices, is a straightforward method which does not need any additional data-structure [9]. Walking performs well in practice for Delaunay triangulations of evenly distributed points [10], but has a non-optimal complexity. Building on the simplicity of the walk, the Jump & Walk [17] and the Delaunay hierarchy [8] improves the complexity while retaining the simplicity of the data-structure. The main idea of these two structures is to find a good starting point for the walk to reduce the number of traversed simplices.

Our Results. This work introduces the *Distribution Condition*: a region $\mathcal{C}$ of a triangulation $\mathcal{T}$ satisfies this condition if the cost of walking in $\mathcal{T}$ along a segment inside $\mathcal{C}$ is proportional to the length of this segment. Then, we relate this condition to the length of the edges of some graphs embedded in $\mathbb{R}^d$, so as to establish complexity results for point location. Section 6 provides experimental evidences that some realistic triangulations verify the Distribution Condition for the whole region inside their *convex-hull*.

We investigate constant-size-memory strategies to choose the starting point of a walk. More precisely, we compare strategies that are dependent on previous queries (self-adapting) and strategies that are not (non-self-adapting), mainly

in the case of random queries. Random queries are, *a priori*, not favorable to self-adapting strategies, since there is no coherence between the queries. Nevertheless, our computations prove that self-adapting strategies are, either better, or not really worse in this case. Thus, there is a good reason for the use of self-adapting strategies since they are competitive even in situations that are seemingly unfavorable. Section 6 provides experiments to confirm such behavior on realistic data.

We revisit Jump & Walk in a self-adapting way by allowing a $\omega(1)$-size memory to choose the starting point for a next query; and give theoretical guarantees that, in this case, the computational complexity is at least the same as the classical Jump & Walk. Section 6 shows that our modification (Keep, Jump, & Walk) has an improved performance compared to the classical Jump & Walk in practice, and it is actually a very competitive method to locate points in a triangulation.

Finally, we show how the Delaunay hierarchy can be used to answer a query q in $O(\log \#(pq))$ randomized expected complexity, where $\#(s)$ indicates the expected number of simplices crossed by the line segment s, and p is a previously located query.

2 Preliminaries

2.1 Previous results on point location

Given a triangulation $\mathcal{T}$ of n points, a query-point q and the location of another point p (starting point), we present here some existing results that will be useful in the sequel.

Walk The query-point q can be located by repeatedly walking from p in $\mathcal{T}$, either from a simplex to its adjacent simplex, or from a vertex to its adjacent vertex, using various strategies [9]. The straight-walk is one of these strategies. It requires visiting the triangles stabbed by a line segment $s = pq$. The straight-walk has a worst-case complexity linear in the number of simplices of the triangulation. If $\mathcal{T}$ is the Delaunay triangulation of points evenly distributed in some finite convex domain and s is not close to the domain boundary, the expected number of simplices stabbed by a segment s is $O(|s| \cdot n^{1/d})$ [10].

Jump & Walk The Jump & Walk technique takes a random sample of k vertices of $\mathcal{T}$, and uses a two-steps location process to locate a query q. First, the jump step determines the nearest vertex in the sample in (brute-force) $O(k)$ time, then a walk in $\mathcal{T}$ is performed from that vertex. Usual analysis of Jump & Walk made the hypothesis that $\mathcal{T}$ is the Delaunay triangulation of points evenly distributed. Taking $k = n^{1/(d+1)}$ gives a final complexity of $O(n^{1/(d+1)})$ [17, 11].

Delaunay hierarchy Building on that idea, the Delaunay hierarchy [8] uses several levels of random samples: At each level of the hierarchy, the walk is performed starting at the closest vertex of the immediately coarser level. Building the hierarchy by selecting a point in the coarser level with some fixed probability, yields a good complexity. In two dimensions, the complexity is $O(\log n)$ in

the worst case. In higher dimensions, this logarithmic time holds if the points are evenly distributed, or even on some weaker hypotheses [8]. We will show in Section 5, that the knowledge of a vertex of the triangulation being not too far from the query, can be used to achieve faster point location.

2.2 Distribution Condition

To analyze the complexity of the straight-walk and derived strategies for point location, we need some hypotheses claiming that the behavior of a walk in a given region $\mathcal{C}$ of the triangulation is as follows.

> *Distribution Condition*: *For a given triangulation $\mathcal{T}$ of some set of points following some distribution in $\mathbb{R}^d$, and a region $\mathcal{C}$ inside the domain of $\mathcal{T}$, there exists a value $\mathcal{F}(\mathcal{T},\mathcal{C}) \in \mathbb{R}$ such that for a segment $s \subseteq \mathcal{C}$, the expected number of simplices of $\mathcal{T}$ intersected by s is less than $1 + \mathcal{F}(\mathcal{T},\mathcal{C}) \cdot |s|$, where the expectation relates to the choice of the sites in the distribution.*

For a fixed region $\mathcal{C}$, if we are considering a distribution of sites such that the part of $\mathcal{T}$ included in $\mathcal{C}$ depends only on the number of points, we will denote $\mathcal{F}(\mathcal{T},\mathcal{C})$ by $\mathcal{F}(n)$, where n is the number of points inside $\mathcal{C}$.

Delaunay triangulations of points following the Poisson distribution in the d-dimensional space assure the Distribution Condition with $\mathcal{F}(n) = O(n^{1/d})$, for any region $\mathcal{C}$. Another example, is the case of points lying on some manifold in a space of dimension d, and we make the following conjecture (supported by our experiments in Section 6):

Conjecture 1. *The Delaunay triangulations under Euclidean metrics of n points evenly distributed on a smooth hypersurface Π of dimension d, verify the Distribution Condition inside $CH(\Pi)$, with $\mathcal{F}(n) = O(n^{1/(d-1)})$. Here, $CH(\Pi)$ is the convex-hull of Π.*

The Distribution Condition affects the relationship between the cost of locating points and the proximity between points. Indeed, the expected cost of locating a finite sequence S of m query-points inside a region $\mathcal{C} \subseteq CH(\mathcal{T})$, given that $\mathcal{C}$ verifies the Distribution Condition for $\mathcal{T}$, is at most

$$\mathcal{F}(\mathcal{T},\mathcal{C}) \cdot \sum_{i=1}^{m} |e_i| + m, \tag{1}$$

where e_i is the line segment formed by the i-th starting point and the i-th query-point.

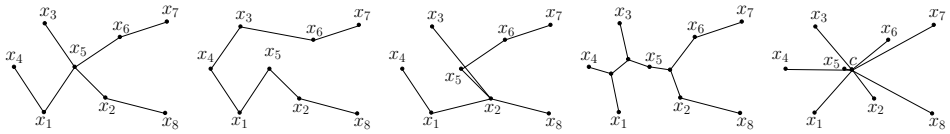
Since the only points we can use as starting points for next locations are points that we know where they are, the line segments $(e_i)_{1 \leq i \leq m}$ must be connected. Therefore the graph $\mathcal{E}$ formed by these line segments is a tree spanning the query-points; such a tree is called the *Location Tree* in the sequel. Its length is given by:

$$|\mathcal{E}| = \sum_{e \in \mathcal{E}} |e|.$$

Note that the Location Tree might have vertices not belonging to S.

2.3 On Trees Embedded in $\mathbb{R}^d$

The tree theory is older than computational geometry itself. Here, we mention some of the well-known trees (and graphs) [22], which are related with the theory of point location. Let $S = \{x_i, 1 \leq i \leq n\}$ be a set of query-points in $\mathbb{R}^d$ and $G = (V, E)$ be the complete graph such that the vertex $v_i \in V$ is embedded on the point $x_i \in S$; the edge $e_{ij} \in E$ linking two vertices v_i and v_j is weighted by its Euclidean length $|x_i - x_j|$. G is usually referred to as the *geometric graph* of S.



(a) $EMST$ (b) $EMLP$ (c) $EMIT$ (d) EST (e) ESS

Figure 1: **Trees embedded in $\mathbb{R}^d$.**

2.3.1 Definitions

We review below some well-known trees. Two special kinds of tree get a special name: (i) a *star* is a tree having one vertex that is linked to all others; and (ii) a *path* is a tree having all vertices of degree 2 but two with degree 1.

EMST. Among all the trees spanning S, a tree with the minimal length is called an *Euclidean minimum spanning tree* of S and denoted $EMST(S)$, see Figure 1a. $EMST$ can be computed with a greedy algorithm at a polynomial complexity.

EMLP. If instead of searching a tree, we search a path with minimal length spanning S, we get the *Euclidean minimum length path* denoted by $EMLP(S)$, see Figure 1b. Another related problem is the search for a minimal tour spanning S: the *Euclidean traveling salesman tour*, denoted by $ETST$. Both problems are NP-complete.

Since a complete traversal of the $EMST$ (either prefix, infix or postfix) produces a tour, and removing an edge of $ETST$ produces a path, we have

$$|EMST(S)| \leq |EMLP(S)| < |ETST(S)| < 2|EMST(S)| \tag{2}$$

EMIT. Above, subgraphs of G are independent of any ordering of the vertices. Now, consider that an ordering is given by a permutation σ, vertices are inserted in the order $v_{\sigma(1)}, v_{\sigma(2)}, \ldots, v_{\sigma(n)}$. We build incrementally a spanning tree T_i for $S_i = \{x_{\sigma(j)} \in S, i \leq j\}$ with $T_1 = \{v_{\sigma(1)}\}$ and $T_i = T_{i-1} \cup \{v_{\sigma(i)}v_{\sigma(j)}\}$, such that $v_{\sigma(i)}v_{\sigma(j)}$ has the shortest length for any $1 \leq j < i$. This tree is called the *Euclidean minimum insertion tree*, and will be denoted by $EMIT(S)$, see Figure 1c. Unlike the previous trees, $EMIT$ does not require points to be known in advance, and hence it is a dynamic structure. Its length depends on σ and for some carefully chosen permutations it coincides with $|EMST|$.

EST. The use of additional vertices usually allows to decrease the length of a tree. Such additional vertices are called *Steiner points* and the minimum-length

tree with Steiner points is the *Euclidean Steiner tree* of S; it is denoted by $EST(S)$, see Figure 1d. Finding EST is NP-complete.
ESS. A star has one vertex linked to all other vertices. If this vertex is an additional vertex that does not belong to V, we can choose its position so as to minimize the length of the star. This point is called the *Fermat-Weber point* of S and the associated star is denoted by $ESS(S)$ (*Euclidean Steiner star*), see Figure 1e.

2.3.2 Results on Tree's Length.

We present here some results on the length of the above-mentioned structures. We start by subgraphs independent of an ordering of the vertices. The Beardwood, Halton and Hammersley theorem [4] states that if x_i are i.i.d. random variables with compact support, then $|ETST(S)| = O(m^{1-1/d})$ with probability 1. By Eq.(2) the same bound is obtained for $|EMLP(S)|$ and $|EMST(S)|$. While this result gives a practical bound on the complexity, they are dependent on probabilistic hypotheses. This was shown to be unnecessary. Steele proves [24] that the complexity of these graphs remains bounded by $O(m^{1-1/d})$ even in the worst case.

Consider the length of the path formed by sequentially visiting each vertex in V. This gives a total length of $\sum_{i=2}^{m} |x_{i-1}x_i|$. Let $V_\sigma = \{x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(m)}\}$ be a sequence of m points made by reordering V with a permutation function σ such that points in V_σ would appear in sequence on some *space-filling* curve. Platzman and Bartholdi [19] proved that in two dimensions the length of the path made by visiting S_σ sequentially is a $O(\log m)$ approximation of $|ETST(S)|$, and hence $\sum_{i=2}^{m} |x_{\sigma(i-1)}x_{\sigma(i)}| = O(\sqrt{m}\log m)$. One of the main interests of such heuristic is that σ can be found in $O(m \log m)$ time.

The asymptotic length of $|EMIT(S)|$, which is dynamic, is shown to be the same as the one of $|EMST(S)|$, which is static. More precisely:

Theorem 2 (Steele [23])**.** *Let S be a sequence of m points in $[0,1]^d$, $d \geq 2$, then we have the following inequality: $|EMST(S)| \leq |EMIT(S)| \leq \gamma_d m^{1-1/d}$. Here, $\gamma_d = 1 + 2^{4d}d^{d/2}/(d-1)$ is a constant depending only on d.*

3 Constant-Size-Memory Strategies

In this section, we analyze the Location Tree length of strategies that store a constant number of possible starting points for a straight-walk. We also provide a comparative study between them.

3.1 Fixed-point strategy.

In the fixed-point strategy, the same point c is used as starting point for all the queries, then the Location Tree is the star rooted at c, denoted by $S_c(S)$. The best Location Tree we can imagine is the Steiner star, but of course computing it is not an option, neither in a dynamic setting nor in a static setting. This strategy is used in practice: In CGAL 3.5, the default starting point for a walk is the so-called *vertex at infinity*; thus the walk starts somewhere on the convex hull, which looks like a kind of worst strategy.

In the worst case, one can easily find a set of query-points S such that $|ESS(S)| = \Omega(m)$, or such that $|S_c(S)|/|ESS(S)|$ goes to infinity for some c. Now we focus on the case of evenly distributed queries.

Theorem 3. *Let S be a sequence of m query-points uniformly i.i.d in the unit ball, then the expected Location Tree length of the best fixed-point strategy is*

$$\left(\frac{d}{d+1}\right) \cdot m.$$

Proof. The proof uses simple integral computations. By symmetry, the Fermat-Weber point goes to the center of the sphere when m goes to infinity, thus we evaluate the average length $E(|Op|)$ between a random point p and the origin. Let $\mathcal{B}_l$ be a ball with radius l centered at the origin, we have

$$E(|Op|) = \int_0^1 l Prob(p \in \mathcal{B}_{l+dl} \setminus \mathcal{B}_l)\mathrm{d}l = \int_0^1 l\frac{(\mathrm{d}V_d(l)/l)\mathrm{d}l}{V_d(1)} = \int_0^1 dl^d \mathrm{d}l = \frac{d}{d+1},$$

where $V_d(l)$ is the volume of a ball of radius l (and $\mathrm{d}V_d(l)/l$ is its area). Multiplying by m gives the expected Location Tree length. □

Theorem 4. *Let S be a sequence of m query-points uniformly i.i.d in the unit ball, then the expected Location Tree length of the worst (on the choice of c inside the ball) fixed-point strategy is*

$$2^{d+1}\left(\frac{2d+1}{2d+2}\right)\frac{B\left(\frac{d}{2}+\frac{1}{2},\frac{d}{2}+1\right)}{B\left(\frac{d}{2}+\frac{1}{2},\frac{1}{2}\right)} \cdot m,$$

where $B(x,y) = \int_0^1 \lambda^{x-1}(1-\lambda)^{y-1}d\lambda$ is the so-called Beta function.

By symmetry, any point on the boundary of the unit sphere is a worse center for a star. The computation of the average is a bit more involved than in Theorem 3, and we split the computation in several lemmas.

Lemma 5. *Consider the spherical cap $\mathcal{H}_h$ formed by crossing a ball $\mathcal{B}_R$ with radius R centered at the origin, with the plane $x = R - h$. Denote h the height of the cap. The volume of $\mathcal{H}_h$ is the volume of the intersection between the half-space $x \geq R - h$ and $\mathcal{B}_R$. This volume is given by:*

$$R^d \frac{\pi^{\frac{d-1}{2}}}{\Gamma(\frac{d+1}{2})} \int_0^{\arccos\left(\frac{R-h}{R}\right)} \sin^d(\lambda)\mathrm{d}\lambda. \tag{3}$$

Proof. The volume $V_d(r)$ of a ball with radius r in dimension d is given by $r^d \cdot \pi^{\frac{d}{2}}/\Gamma(1+\frac{d}{2})$. Each cross-section $x = R - h + \delta$, $0 \leq \delta \leq h$ is a $(d-1)$-dimensional ball. If we integrate all those balls along the x axis, we have $\int_{R-h}^{R} V_{d-1}(\sqrt{R^2 - t^2})\mathrm{d}t$. Eq.(3) follows from replacing t by $\lambda = R\cos(t)$. □

Lemma 6. *Let Ω be a point on the boundary of the unit ball $\mathcal{B}_{unit}$, and $P_{\mathcal{H}}(l) = Prob(|\Omega p| \leq l\,; p \in \mathcal{B}_{unit})$ be the cumulative distribution function of distances between an uniformly distributed random point inside $\mathcal{B}_{unit}$ and Ω, then*

$$P_{\mathcal{H}}(l) = \frac{1}{B\left(\frac{d}{2}+\frac{1}{2},\frac{1}{2}\right)}\left(\int_0^{\arccos(1-l^2/2)} \sin^d(\lambda)\mathrm{d}\lambda + l^d \int_0^{\arccos(l/2)} \sin^d(\lambda)\mathrm{d}\lambda\right),$$

where $B(x,y) = \int_0^1 \lambda^{x-1}(1-\lambda)^{y-1}\mathrm{d}\lambda$ is the Beta function.

Proof. If we denote $\mathcal{B}_l$ the ball of radius l centered in Ω, the desired probability is clearly $volume(\mathcal{B}_l \cap \mathcal{B}_{unit})/volume(\mathcal{B}_{unit})$. $\mathcal{B}_l \cap \mathcal{B}_{unit}$ is the union of two spherical caps limited by the plane $x = 1 - l^2/2$ which can be computed using Lemma 5. □

Proof of Theorem 4. The theorem follows from:

$$\begin{aligned} E(|\Omega p|) &= \int_0^2 l P'_{\mathcal{H}}(l) dl \\ &= \int_0^2 l \left(\frac{\frac{1}{2} l^d \left(1 - \frac{l^2}{4}\right)^{\frac{d-1}{2}}}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)} + d l^{d-1} \frac{\int_0^{\arccos(l/2)} \sin^d(\lambda) \mathrm{d}\lambda}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)} \right) \mathrm{d}l \\ &= \frac{1}{2} \int_0^2 \frac{l^{d+1} \left(1 - \frac{l^2}{4}\right)^{\frac{d-1}{2}}}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)} \mathrm{d}l + \frac{1}{2} \int_0^2 2 d l^d \frac{\int_0^{\arccos(l/2)} \sin^d(\lambda) \mathrm{d}\lambda}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)} \mathrm{d}l. \end{aligned} \quad (4)$$

The right part of Expression (4) corresponds exactly to the expected value of l where l is the length of a random segment determined by two evenly distributed points in the unit ball [21]. Its value is given by:

$$\int_0^2 2 d l^d \frac{\int_0^{\arccos(l/2)} \sin^d(\lambda) \mathrm{d}\lambda}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)} \mathrm{d}l = 2^{d+1} \left(\frac{d}{d+1}\right) \frac{B\left(\frac{d}{2} + \frac{1}{2}, \frac{d}{2} + 1\right)}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)}. \quad (5)$$

The left part of Expression (4) can be obtained as follows:

$$\begin{aligned} \int_0^2 \frac{l^{d+1} \left(1 - \frac{l^2}{4}\right)^{\frac{d-1}{2}}}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)} \mathrm{d}l &= 2 \int_0^1 \frac{2^{d+1} y^{d+1} \left(1 - y^2\right)^{\frac{d-1}{2}}}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)} \mathrm{d}y \\ &= \int_0^1 \frac{2^{d+1} z^{\frac{d}{2}} (1 - z)^{\frac{d-1}{2}}}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)} \mathrm{d}z \\ &= 2^{d+1} \frac{B\left(\frac{d}{2} + \frac{1}{2}, \frac{d}{2} + 1\right)}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)}. \end{aligned}$$

Finally, we have

$$E(|\Omega p|) = 2^{d+1} \left(\frac{2d+1}{2d+2}\right) \frac{B\left(\frac{d}{2} + \frac{1}{2}, \frac{d}{2} + 1\right)}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)}.$$

Multiplying by m gives the expected Location Tree length. □

Corollary 7. *Let S be a sequence of m query-points uniformly i.i.d in the unit ball, then the ratio between the expected Location Tree lengths of the best and worst fixed-point strategies is at most 2 (for $d = 1$), and at least $\sqrt{2}$ (when $d \to \infty$).*

Proof. This ratio is a decreasing function of the dimension. Using Theorem 3 and Theorem 4, the ratio $\rho(d)$ between the expected Location Tree length of the worst and the best fixed-point strategies is given by:

$$\rho(d) = 2^{d+1} \left(\frac{2d+1}{2d} \right) \frac{B\left(\frac{d}{2}+1, \frac{d}{2}+\frac{1}{2}\right)}{B\left(\frac{d+1}{2}, \frac{1}{2}\right)} \tag{6}$$

Since $\rho(d)$ is a monotonic decreasing function, its extremal values are $\rho(1)$ and $\lim_{d\to\infty} \rho(d)$. We have trivially from Eq.(6) that $\rho(1) = 2$. To prove Corollary 7, it remains to find $\lim_{d\to\infty} \rho(d)$. Using the Stirling's identities:

$$\begin{aligned} B(a,b) &\sim \sqrt{2\pi} \tfrac{a^{a-\frac{1}{2}} b^{b-\frac{1}{2}}}{(a+b)^{a+b-\frac{1}{2}}}, & a,b \gg 0, \\ B(a,b) &\sim \Gamma(b) a^{-b}, & a \gg b > 0, \end{aligned}$$

we have:

$$\begin{aligned} \lim_{d\to\infty} \rho(d) &= \lim_{d\to\infty} 2^{d+1} \left(\frac{2d+1}{2d} \right) \frac{B\left(\frac{d}{2}+1, \frac{d}{2}+\frac{1}{2}\right)}{B\left(\frac{d+1}{2}, \frac{1}{2}\right)} \\ &= \lim_{d\to\infty} 2^{d+1} \frac{B\left(\frac{d}{2}+1, \frac{d}{2}+\frac{1}{2}\right)}{B\left(\frac{d+1}{2}, \frac{1}{2}\right)} \\ &= \lim_{d\to\infty} \frac{2^{d+1}\sqrt{2\pi}\left(\frac{d}{2}+\frac{1}{2}\right)^{\frac{d}{2}} \left(\frac{d}{2}+1\right)^{\frac{d}{2}+\frac{1}{2}}}{\sqrt{\pi}\left(d+\frac{3}{2}\right)^{d+1} \left(\frac{d}{2}+\frac{1}{2}\right)^{-\frac{1}{2}}} \\ &= 2^{1/2} \cdot \lim_{d\to\infty} \frac{(d+1)^{\frac{d+1}{2}} (d+2)^{\frac{d+1}{2}}}{(d+\frac{3}{2})^{\frac{d+1}{2}} (d+\frac{3}{2})^{\frac{d+1}{2}}} \\ &= 2^{1/2} \cdot e^{-\frac{1}{4}} \cdot e^{\frac{1}{4}} \\ &= \sqrt{2} \end{aligned}$$

Then we have that $\sqrt{2} \leq \rho(d) \leq 2$ for $d \geq 1$. □

Figure 2 gives the expected average length of an edge of the best and worst fixed-point Location Trees.

3.2 Last-point strategy.

An easy alternative to the fixed-point strategy is the last-point strategy. To locate a new query-point, the walk starts from the previously located query. The Location Tree obtained with such a strategy is a path. When $\mathcal{T}$ verifies the Distribution Condition, the optimal path is the $EMLP(S)$.

In the worst case, the length of such a path is clearly $\Omega(m)$; an easy example is to repeat alternatively the two same queries. In contrast with the fixed-point strategy, the last-point strategy depends on the query distribution. If the queries have some spatial coherence, it is clear that we improve on the fixed-point strategy. Such a coherence may come from the application, or by reordering the queries. There is always a permutation of indices on S such that the total length of the path is sub-linear [24, 12]. Furthermore, in two dimensions, one could find such permutation in $O(m \log m)$ time complexity [19].

Now, the question is "if there is no spatial coherence, how the fixed and last point strategies do compare?".

Theorem 8. *The ratio between the Location Tree lengths of the last-point strategy and the fixed-point strategy is at most* 2*.*

Proof. This is an easy consequence of the triangle inequality.
Take $S = x_1, x_2, \ldots, x_m$, and any fixed-point c. Then we have:

$$|x_i x_{i+1}| \leq |cx_i| + |cx_{i+1}|,$$

for all $1 \leq i < m$. Summing the term above for each value of i leads to the inequality:

$$\sum_{i=1}^{m-1} |x_i x_{i+1}| \leq \sum_{i=1}^{m-1} |cx_i| + \sum_{i=2}^{m} |cx_i| \leq 2 \sum_{i=1}^{m} |cx_i|,$$

which completes the proof. □

Theorem 9. *The ratio between the Location Tree lengths of the last-point strategy and the fixed-point strategy is arbitrarily small.*

Proof. Consider a set of m queries distributed on a circle in $\mathbb{R}^d$. If the queries are visited along the circle, the length of the location tree of the last-point strategy is $O(1)$, while $|ESS| = \Omega(m)$. □

Combining the results in Theorem 8 and Theorem 9, it is reasonable to conclude that the last-point strategy is better in general, as the improvement the fixed-point strategy could bring does not pay the price of its worst-case behavior. We now study the case of evenly distributed queries.

Theorem 10. *Let S be a sequence of m query-points uniformly i.i.d in the unit ball, then the expected Location Tree length of the last-point strategy is*

$$2^{d+1} \left(\frac{d}{d+1} \right) \frac{B\left(\frac{d}{2} + \frac{1}{2}, \frac{d}{2} + 1\right)}{B\left(\frac{d}{2} + \frac{1}{2}, \frac{1}{2}\right)} \cdot m.$$

Proof. This is equivalent to find the expected length of a random segment determined by two evenly distributed points in the unit ball, which is given in [21] for instance. □

Theorems 3, 4, and 10 give the following result:

Corollary 11. *Let S be a sequence of m query-points uniformly i.i.d in the unit ball, then the ratio between the expected Location Tree lengths of the last-point and the best fixed-point strategies is at most $\sqrt{2}$ (when $d \to \infty$), and at least $4/3$ (when $d = 1$) whereas the ratio between the expected Location Tree lengths of the last-point and the worst fixed-point strategies is $2d/(2d+1)$.*

As shown in Figure 2, the last-point strategy is in between the best and worst fixed-point strategies, but closer and closer to the worst one when d increases. Thus, in the context of evenly distributed points in a ball, the last-point strategy cannot be worse than any fixed point strategy by more than a factor of $\sqrt{2}$. Still, the fixed-point strategy may have some interests under some conditions: (i) queries are known *a priori* to be random **and**; (ii) a reasonable approximation of the center of $ESS(S)$ can be found.

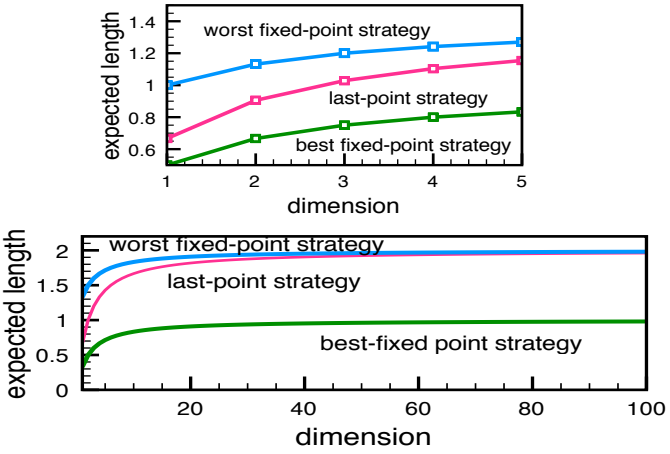


Figure 2: **Expected lengths.** *Expected average lengths of an edge of the last-point, best and worst fixed-point Location Trees. The domain $\mathcal{C}$ here is a d-dimensional ball, and the queries are evenly distributed in $\mathcal{C}$.*

Theorem 10 and Corollary 11 assume that the region $\mathcal{C}$, where the queries lie in, is a ball. Now, one might ask whether the shape of $\mathcal{C}$ affects the cost of the strategies. In the quest for an answer, we may consider query-points uniformly i.i.d in the unit cube $[0,1]^d$. This leads to the following related expressions:

$$\begin{aligned} B_d &= \int_0^1 \dots \int_0^1 \left(\lambda_1^2 + \dots + \lambda_d^2\right)^{1/2} \mathrm{d}\lambda_1 \dots \mathrm{d}\lambda_d, \\ X_d &= \int_0^1 \dots \int_0^1 \left((\lambda_1 - 1/2)^2 + \dots + (\lambda_d - 1/2)^2\right)^{1/2} \mathrm{d}\lambda_1 \dots \mathrm{d}\lambda_d, \\ \Delta_d &= \int_0^1 \dots \int_0^1 \left((\lambda_1 - \lambda'_1)^2 + \dots + (\lambda_d - \lambda'_d)^2\right)^{1/2} \mathrm{d}\lambda_1 \dots \mathrm{d}\lambda_d \, \mathrm{d}\lambda'_1 \dots \mathrm{d}\lambda'_d, \end{aligned}$$

where B_d, X_d, and Δ_d are respectively the average length of an edge of: the largest star (rooted at a corner), the smallest star (rooted at the center), and of a random path. Above expressions are often referred to as *box integrals* [3]. First, note that by substitution of variable we have $B_d/X_d = 2$, independently of d. In Anderssen *et al.* [20], we have that, $X_d \sim \sqrt{d/3}$ and $\Delta_d \sim \sqrt{d/6}$ and thus B_d/Δ_d and $\Delta_d/X_d \sim \sqrt{2}$ when d goes to infinity. These ratios have to be compared with Corollary 11. If $\mathcal{C}$ is an unit cube, the expected Location Tree length of the last-point, best and worst fixed-point strategies remains in bounded ratio, but with different values compared to the case where $\mathcal{C}$ is a ball. Notice however that, when d is large, the ratio between the best fixed-point and the last-point strategies remains $\sqrt{2}$ in both cases. This ratio in some sense is more robust than the ratios involving the worst-case fixed-point strategy.

3.3 k-last-points strategy.

We explore here a variation of the last-point strategy. Instead of remembering the place where the last query was located, we store the places where the k last queries were located, for some small constant k. These k places are called *landmarks* in what follows. Then to process a new query, the closest landmarks

are determined by $O(k)$ brute-force comparisons, then a walk is performed from there. This strategy has some similarity with Jump & Walk, the main differences are that the sample has fixed size and depends on the query distribution (it is dynamically modified).

The Location Tree associated with such a strategy has bounded degree $k+1$ (or the *kissing number* in dimension d, if it is smaller than $k+1$) and its length is greater than $|EMST|$ and smaller than the length of the path associated to the same vertices ordering, thus previous results provide upper and lower bounds. A tree of length $\Omega(m/k) = \Omega(m)$ is easily achieved by repeating a sequence of k queries along a circle of length 1. The following Theorem gives the complexity when the queries are evenly distributed:

Theorem 12. *Let S be a sequence of m query-points uniformly i.i.d in the unit ball, then the expected Location Tree length of the k-last-points strategy verifies*

$$\left(\frac{1}{d}\right) B\left(k+1, \frac{1}{d}\right) \cdot m \leq E(length) \leq 2\left(\frac{1}{d}\right) B\left(k+1, \frac{1}{d}\right) \cdot m. \qquad (7)$$

First we will evaluate the distance between the origin and the closest amongst k points $(p_1 p_2, \ldots, p_k)$ evenly distributed in the unit ball.

Lemma 13. *Let $P_{\mathcal{B},k}(l) = Prob\left(\min(|Op_j|)_{1\leq j\leq k} \leq l\right)$ be the cumulative distribution function of the minimum distance among k points following a uniformly i.i.d inside the unit ball, and the center of the unit ball, then*

$$P_{\mathcal{B},k}(l) = 1 - (1 - l^d)^k.$$

Proof.

$$\begin{aligned} P_{\mathcal{B},k}(l) &= Prob\left(\min(|Op_j|)_{1\leq j\leq k} \leq l\right) \\ &= 1 - Prob\left(|Op_j| > l\ , 1 \leq j \leq k\right) \\ &= 1 - Prob\left(|Op_1| > l\right)^k \\ &= 1 - \left(1 - Prob\left(|Op_1| \leq l\right)\right)^k \\ &= 1 - \left(1 - l^d\right)^k, \end{aligned}$$

since $Prob\left(|Op_1| \leq l\right)$ is the ratio of the ball of radius l over the unit ball. □

Lemma 14. *The expected value $E\left(\min(|Op_j|)_{1\leq j\leq k}\right)$ of the minimum distance among k points following a uniformly i.i.d inside the unit ball and the center of the unit ball, is given by*

$$E\left(\min(|Op_j|)_{1\leq j\leq k}\right) = \frac{1}{d} B\left(k+1, \frac{1}{d}\right).$$

Proof. Using Lemma 13, we have:

$$\begin{aligned} E\left(\min(|Op_j|)_{1\leq j\leq k}\right) &= \int_0^1 l P'_{\mathcal{B},k}(l) \mathrm{d}l \\ &= kd \int_0^1 l^d \left(1 - l^d\right)^{k-1} \mathrm{d}l \\ &= k \int_0^1 \lambda^{1/d} \left(1 - \lambda\right)^{k-1} \mathrm{d}\lambda \\ &= kB\left(k, 1 + \frac{1}{d}\right) = \frac{1}{d} B\left(k+1, \frac{1}{d}\right). \end{aligned}$$

□

Lemma 14 gives us the lower-bound in Theorem 12. Now, we shall obtain the upper-bound, which is a bit more involved. First we will evaluate the distance between a point on the boundary and the closest amongst k points $(p_1 p_2, \ldots, p_k)$ evenly distributed in the unit ball.

Lemma 15. *Let Ω be a point on the boundary of the unit ball and $P_{\mathcal{H},k}(l) = Prob\left(\min(|\Omega p_j|)_{1\leq j\leq k} \leq l\right)$ be the cumulative distribution function of the minimum distance among k points following a uniformly i.i.d inside the unit ball, and Ω, then*

$$P_{\mathcal{H},k}(l) = 1 - \left(1 - P_{\mathcal{H}}(l)\right)^k.$$

Proof.

$$\begin{aligned} P_{\mathcal{H},k}(l) &= Prob\left(\min(|\Omega p_j|)_{1\leq j\leq k} \leq l\right) \\ &= 1 - Prob\left(|\Omega p_j| > l\ , 1 \leq j \leq k\right) \\ &= 1 - Prob\left(|\Omega p_1| > l\right)^k \\ &= 1 - \left(1 - Prob\left(|\Omega p_1| \leq l\right)\right)^k, \end{aligned}$$

$P_{\mathcal{H}}(l)$ as given at Lemma 6. The exact expression of $P_{\mathcal{H},k}(l)$ is not necessary in the sequel. □

Now, we obtain a general expression for the expected value of $\min(|\Omega p_j|)_{1\leq j\leq k}$.

Lemma 16. *The expected value $E\left(\min(|\Omega p_j|)_{1\leq j\leq k}\right)$ of the minimum distance among k points following a uniformly i.i.d inside the unit ball and Ω, is given by*

$$E\left(\min(|\Omega p_j|)_{1\leq j\leq k}\right) = \int_0^2 (1 - P_{\mathcal{H}}(l))^k \mathrm{d}l.$$

Proof. As $P_{\mathcal{H}}(0) = 0$ and $P_{\mathcal{H}}(2) = 1$, integration by parts gives us the following identity:

$$\int_0^2 l P'_{\mathcal{H}}(l) P^{i-1}_{\mathcal{H}}(l) \mathrm{d}l = \frac{2 - \int_0^2 P^i_{\mathcal{H}}(l)\mathrm{d}l}{i}, i > 0. \tag{8}$$

We also have the following expression for $E\left(\min(|\Omega p_j|)_{1\leq j\leq k}\right)$:

$$\begin{aligned} E\left(\min(|\Omega p_j|)_{1\leq j\leq k}\right) &= \int_0^2 k l P'_{\mathcal{H}}(l)(1 - P_{\mathcal{H}}(l))^{k-1}\mathrm{d}l \\ &= \sum_{i=0}^{k-1} (-1)^i \binom{k-1}{i} \int_0^2 k l P'_{\mathcal{H}}(l) P^i_{\mathcal{H}}(l)\mathrm{d}l. \end{aligned} \tag{9}$$

Replacing Eq. (8) in Eq. (9) leads to:

$$\begin{aligned} \sum_{i=0}^{k-1} (-1)^i \binom{k-1}{i} \int_0^2 k l P'_{\mathcal{H}}(l) P^i_{\mathcal{H}}(l)\mathrm{d}l &= \sum_{i=0}^{k} (-1)^i \binom{k}{i} \int_0^2 P^i_{\mathcal{H}}(l)\mathrm{d}l \\ &= \int_0^2 \sum_{i=0}^{k} (-1)^i \binom{k}{i} P^i_{\mathcal{H}}(l)\mathrm{d}l \\ &= \int_0^2 (1 - P_{\mathcal{H}}(l))^k \mathrm{d}l. \end{aligned}$$

□

Proof of Theorem 12. Now, if we take a function $\Psi(l)$ such that $\Psi(l) \leq P_{\mathcal{H}}(l)$ for $0 \leq l \leq 2$, it upper-bounds the integral in Lemma 16. Take $\Psi(l) = (l/2)^d$, then we have:

$$\begin{aligned} E\left(\min(|\Omega p_j|)_{1\leq j\leq k}\right) = \int_0^2 (1 - P_{\mathcal{H}}(l))^k \mathrm{d}l &\leq \int_0^2 (1 - \Psi(l))^k \mathrm{d}l \\ &= \int_0^2 \left(1 - \frac{l^d}{2^d}\right)^k \mathrm{d}l \\ &= 2\int_0^1 (1 - \lambda^d)^k \mathrm{d}\lambda \\ &= 2\left(\frac{1}{d}\right) B\left(k+1, \frac{1}{d}\right). \end{aligned}$$

Multiplying by m completes the proof. □

Remark: If we no longer consider k as a constant, then taking $k = m$ makes the Location Tree of the k-last-points strategy an $EMIT$. And hence, using the Stirling's identity $B(x, y) \sim \Gamma(y)x^{-y}$, gives an expected length $|EMIT| = \Theta\left(m^{1-1/d}\right)$. Comparing to Theorem 2, the asymptotic growth in both random and worst case are the same, but the constant is much better in the random case. ■

Intuitively, if the queries have some not too strong spatial coherence, the k-last-points strategy seems a good way to improve the last-point strategy. Surprisingly, experiments in Section 6 shows that even if the points have some strong coherence, a small k strictly greater than 1 improves on the last-point strategy when points are sorted along a space-filling curve. More precisely, $k = 4$ improves the location time by up to 15% on some data sets.

To conclude this section, assume that, in Theorems 3, 4, 10, and 12, the region $\mathcal{C}$ where the m queries lie in satisfies the Distribution Condition for a triangulation $\mathcal{T}$, and the size of the Location Tree is given respectively by L_1, L_2, L_3, and L_4. Then, by Eq.(1), the expected cost to locate the queries in $\mathcal{T}$ is bounded respectively by $m + L_{\{1,2,3,4\}} \cdot \mathcal{F}(\mathcal{T}, \mathcal{C})$.

4 Jump & Walk revisited: Keep, Jump & Walk

In this section, the k-last-points strategy is extended to a variable k. Here, it is necessary to have a closer look at the way of managing the landmarks. The classical Jump & Walk [1] strategy [11, 17] uses a set of k landmarks randomly chosen in the vertices of $\mathcal{T}$, then a query is located by walking from the closest landmark. To ensure adaptation of the query distribution we have several possibilities:
(i) we can use k queries chosen at random in previous queries, (ii) we can use the k last queries for the set of landmarks, and (iii) we can keep all the queries as landmarks, and regularly clear the landmarks set after a batch of k queries.

[1] Apropos, Jump & Walk is a bit confusing terminology, since k is usually chosen such that the jump and the walk take the same time; it does not really match the intuitive idea of the relative speed of a jumper and a walker.

For any rule to construct the set of landmarks, the time to process a query q splits in:
— Keep: the time $Q(k)$ for updating the set of landmarks if needed,
— Jump: the time $J(k)$ for finding the closest landmark l_q, and
— Walk: the time $O(|ql_q|\mathcal{F}(\mathcal{T},\mathcal{C}))$ to walk in the region $\mathcal{C}$ of $\mathcal{T}$, assuming that $\mathcal{C}$ satisfies the Distribution Condition.

Combining various options for $\mathcal{F}(\mathcal{T},\mathcal{C})$ and the data-structure to store the landmarks, gives us some interesting possibilities. The trick is always to balance these different costs, since increasing one decreases another.

Jump & Walk. Classical Jump & Walk uses a simple data-structure (e.g. a list) to store the random sample of $\mathcal{T}$ and assumes $\mathcal{F}(n) = O(n^{1/d})$. Here, we will use the same data-structure to store the set of landmarks. Keep step decides whether the query is kept at a landmark and inserts it if needed. This takes $Q(k) = O(1)$. Jump step takes $J(k) = O(k)$. Then taking $k = n^{1/(d+1)}$ landmarks amongst the queries ensures an amortized query time of $O(n^{1/(d+1)})$ as $|ql_q| = O(k^{-1/d})$ by Theorem 2. It is noteworthy that the complexity obtained here matches the classical Jump & Walk complexity with no hypotheses on the distribution of query-points (naturally, the queries must lie in the region $\mathcal{C}$, which in turn must lie inside the domain of $\mathcal{T}$, see Section 2.2).

Outside this classical framework, Jump & Walk has some interests, even with weaker hypotheses. Theorem 2 ensures that $|ql_q|$ has amortized length $O(k^{-1/d})$. Therefore, taking $k = \mathcal{F}(\mathcal{T},\mathcal{C})^{1-1/(d+1)}$ balances the jump and the walk costs. Another remark is that if the landmarks are a random subset of the vertices of $\mathcal{T}$ (as is the classical Jump & Walk), then the cost of the walk is $\mathcal{F}(n/k)$ [8, Variation of Lemma 4]. Assuming $\mathcal{F}(j) = O(j^\beta)$, the jump and the walk costs are balanced by taking $k = n^{1-1/(\beta+1)}$ in this case.

If Conjecture 1 is verified, Jump & Walk should use a sample of size $O(n^{3/8})$ to construct Delaunay triangulation for surface reconstruction purpose, and not $O(n^{1/4})$ as for random points in 3D; this is verified experimentally in Section 6.

Walk & Walk. In Walk & Walk, the data-structure to store the landmarks is a Delaunay triangulation $\mathcal{L}$, in which it is possible to walk (notice that $\mathcal{T}$ may not be a Delaunay triangulation). Assuming a random order on the landmarks, inserting or deleting a landmark after location takes $Q(k) = O(1)$ and jump step takes $J(k) = O(\mathcal{F}(\mathcal{L},\mathcal{C}))$.

If the queries and the sites are both evenly distributed we get $J(k) = O(k^{1/d})$ and $|ql_q| \cdot \mathcal{F}(\mathcal{T},\mathcal{C}) = O(k^{-1/d} \cdot n^{1/d})$ which gives $k = \sqrt{n}$ to balance the jump and walk costs. Finally, the point location takes expected time $O(n^{1/2d})$.

If walking inside $\mathcal{T}$ and $\mathcal{L}$ takes linear time, $k = n^{1-1/(d+1)}$ balances Walk & Walk costs.

Delaunay Hierarchy of Queries. A natural idea is to use several layers of triangulations, walking at each level from the location at the coarser layer. When the landmarks are vertices of $\mathcal{T}$ and each sample takes a constant ratio of the vertices at the level below, this idea yields the Delaunay hierarchy [8].

Storing the queries in a Delaunay hierarchy may have some interesting effects: If the region $\mathcal{C}$ of $\mathcal{T}$ has some bad behavior $\mathcal{F}(\mathcal{T},\mathcal{C}) \gg n^{1/d}$ and there is many well-distributed queries, we can get interesting query time to the price of polynomial storage. More precisely, if the queries are such that a random sample of the queries has a Delaunay triangulation of expected linear size (always true in 2D), then using a random sample of k queries for the landmarks and a

Delaunay hierarchy to store $\mathcal{L}$, gives $Q(k) = J(k) = O(\log k)$. Then by Theorem 2 we have $|ql_q| = O(k^{-1/d})$ (amortized) and taking $k = \mathcal{F}(\mathcal{T},\mathcal{C})^d/\log^d n$ balances jump and walk costs, giving a logarithmic location time.

5 Climbing up in the Delaunay Hierarchy

Up to now, the aim was to choose a good starting point to walk in $\mathcal{T}$. In this section, we show how a good starting point can be used within the Delaunay hierarchy to improve point location. Assume $\mathcal{T}$ is a Delaunay triangulation, then classical use of the Delaunay hierarchy provides a logarithmic cost in the total size of $\mathcal{T}$ to locate a point. The cost we reach here is logarithmic in the local complexity of the triangulation, that is logarithmic in the number of vertices of $\mathcal{T}$ *in between* the starting point and the query.

Given a set of n points $\mathcal{P}$ in the plane, the Delaunay hierarchy [8] constructs random samples $\mathcal{P} = \mathcal{P}_0 \subseteq \mathcal{P}_1 \subseteq \mathcal{P}_2 \subseteq \ldots \subseteq \mathcal{P}_h$ such that $Prob(p \in \mathcal{P}_{i+1} \,|\, p \in \mathcal{P}_i) = 1/\alpha$ for some constant $\alpha > 1$. The $h+1$ Delaunay triangulations $\mathcal{D}_i$ of $\mathcal{P}_i$ are computed and the hierarchy is used to find the nearest-neighbor of a query q by walking at one level i from the nearest-neighbor of q at the level $i+1$. It is proven that the expected cost of walking at one level is $O(\alpha)$ and since the expected number of levels is $\log_\alpha n$, we obtain a logarithmic expected time to descend the hierarchy for point location.

If a good starting vertex $v = v_0$ in $\mathcal{D}_0$ is known, the Delaunay hierarchy can be used in another way: From v_0 a walk starts in $\mathcal{D}_0$ visiting simplices crossed by segment v_0q; the walk is stopped, either if the simplex containing q is found, or if a simplex having a vertex v_1 belonging to the sample $\mathcal{P}_1$ is found. If the walk stops because v_1 is found, then a new walk in $\mathcal{D}_1$ starts at v_1 along segment v_1q. This process continues recursively up to the level l, where a simplex of $\mathcal{D}_l$ that contains q is found. Finally, the hierarchy is descended as in the usual point location.

Theorem 17. *Given a set of n points $\mathcal{P}$, and a convex region $\mathcal{C} \subseteq CH(\mathcal{P})$, such that $\mathcal{C}$ satisfies the Distribution condition for the Delaunay triangulation of a random sample of size r of $\mathcal{P}$ with $\mathcal{F}(r)$ polynomial, then the expected cost of climbing and descending the Delaunay hierarchy from a vertex v to a query point q, both lying in $\mathcal{C}$, is $O(\log w)$, where w is the cost of the walk from v to q in $\mathcal{D}$ the Delaunay triangulation of $\mathcal{P}$.*

Proof. **Climbing one level.** Since the probability that any vertex of $\mathcal{D}_i$ belongs to $\mathcal{D}_{i+1}$ is $1/\alpha$, and that each time a new simplex is visited during the walk a new vertex is discovered, the expected number of visited simplices before the detection of a vertex that belongs to $\mathcal{D}_{i+1}$ is $1 + \sum_{j=0}^{\infty} j\frac{1}{\alpha}\left(1-\frac{1}{\alpha}\right)^j = \alpha$.
Descending one level. The cost of descending one level is $O(\alpha)$ [8, Lemma 4].
Number of levels. Let w_i denote the number of edges crossed by v_iq in $\mathcal{D}_i$; the Distribution Condition gives $w_i = \mathcal{F}(n/\alpha^i)|v_iq| \leq \mathcal{F}(n/\alpha^i)|v_0q|$. If $\mathcal{F}(r)$ is a polynomial function $O(r^\beta)$, the expected number of levels that we climb before descending is less than $l = (\log w_0)/\beta$, since we have

$$w_l = \mathcal{F}(n/\alpha^l)|v_lq| \leq \mathcal{F}(n/\alpha^l)|v_0q| = w_0/\alpha^{l\beta} = w_0/\alpha^{\log w_0} = 1$$

(where the big O have been omitted). Thus, at level l the walk takes constant time. □

6 Experimental Results

Experiments[2] have been realized on synthetic and realistic models[3] (scaled to fit in the unit cube).

6.1 The Distribution Condition

Our first set of experiments is an experimental verification of the Distribution Condition. We compute the Delaunay triangulation of different inputs, either artificial or realistic, with several sizes; for realistic inputs we construct files of various sizes by taking random samples of the desired size.

We consider several data sets in 3D:
— points distributed in a cube with random uniform distribution,
— points distributed in a cube with a $\rho = x^2$ density,
— points distributed on the surface of an ellipsoid with random uniform distribution, the lengths of the ellipsoid axes are 1/3, 2/3, and 1,
— Pooran's Hand is a data set obtained by scanning a 3D model of a Hand, and
— Galaad is a data set obtained by scanning a 3D model of a toy soldier.

Files of different sizes, smaller than the original model are obtained by taking random samples of the main file with the desired number of points.

Pooran's Hand and Galaad are originally defined in a parallelepiped much bigger than $[0,1]^3$. We scaled each axis by the same factor and translated so to have these models constrict inside $[0,1]^3$.

Figure 3 shows the number of simplices crossed by the walk in terms of the length of the walk, for various, randomly chosen, walks in the triangulation. Notice that even if there is some dispersion in the result, the dispersion is much more below than above the clouds of points. This is a very good news because it means that you more likely to go faster than slower compared to the expected behavior.

From that picture, the slope of lines through these points give $\mathcal{F}(\mathcal{T},\mathcal{CH}(\mathcal{T}))$, then we draw $\mathcal{F}(\mathcal{T},\mathcal{CH}(\mathcal{T}))$ in terms of the triangulation size in Figure 4; the slope of the different curves gives the exponent of n. The points sampled on an ellipsoid give $\log \mathcal{F}(n) \sim 0.52 \log n$, which is not far from Conjecture 1 that claims $\mathcal{F}(n) = O(n^{1/2})$. The points evenly distributed in a cube gives $\log \mathcal{F}(n) \sim 0.31 \log n$, which is not far from $\mathcal{F}(n) = O(n^{1/3})$.

6.2 k-last-points strategy

CGAL library [18] uses spatial sorting [6] to introduce a strong spatial coherence in a set of points. For several models, we locate $1M$ queries evenly distributed

[2] The hardware used for the experiments described in the sequel, is a MacBook Pro 3,1 equipped with an 2.6 GHz Intel Core 2 processor and 2 GB 667 MHz DDR2 SDRAM, Mac OS X version 10.5.7. The software uses CGAL 3.5 [5] and is compiled with `g++ 4.3.2` and options `-O3 -DNDEBUG`.

[3] The scanned models used here: Pooran's Hand and Galaad, are taken from Aim@shape repository.

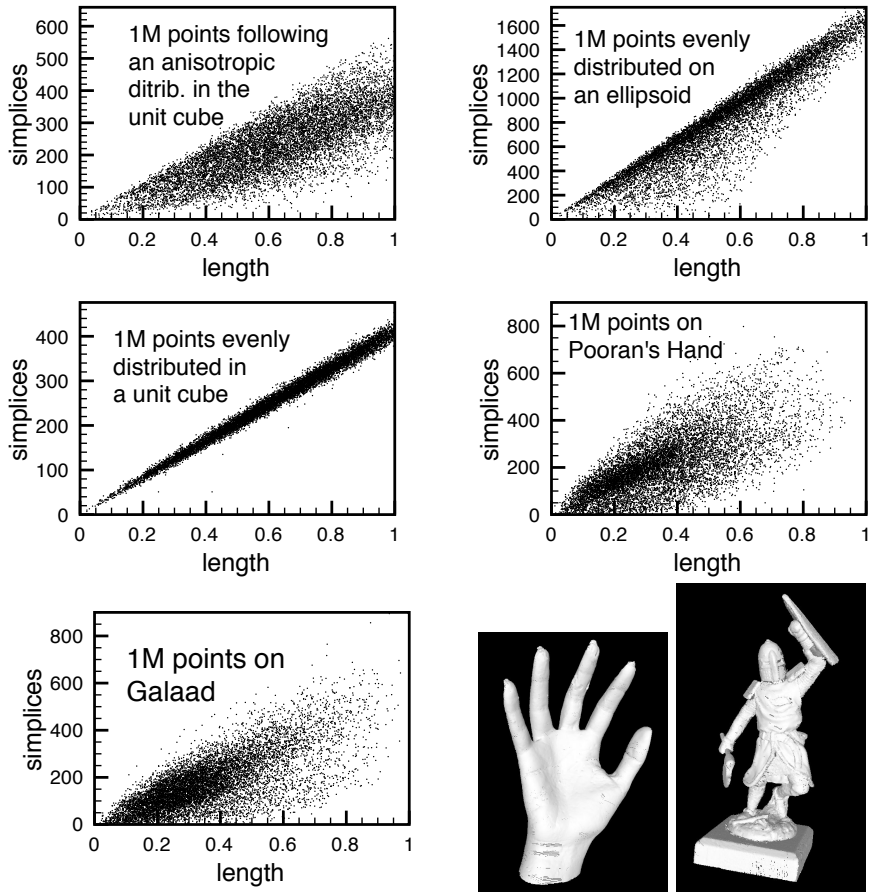


Figure 3: **Distribution condition.** ♯ *of crossed tetrahedra in terms of the length of the walk.*

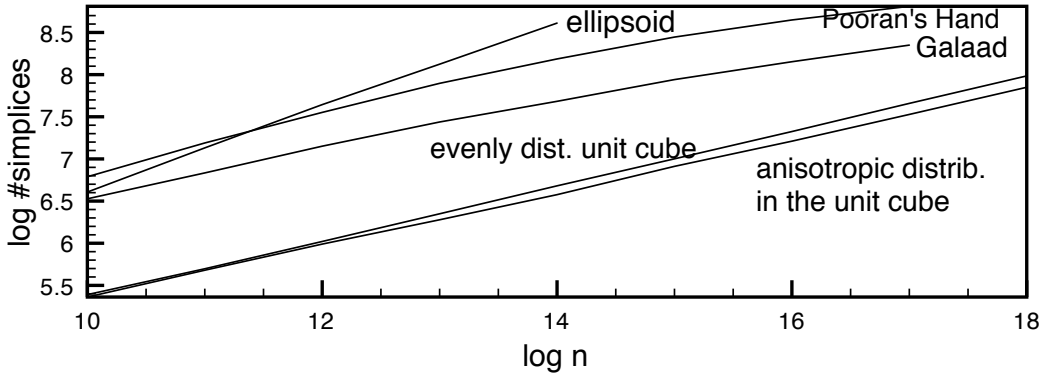


Figure 4: **Distribution condition.** *Assuming $\mathcal{F}(n) = n^\alpha$, α is given by the slope of above "lines" (*log *here is* $\log_2$*).*

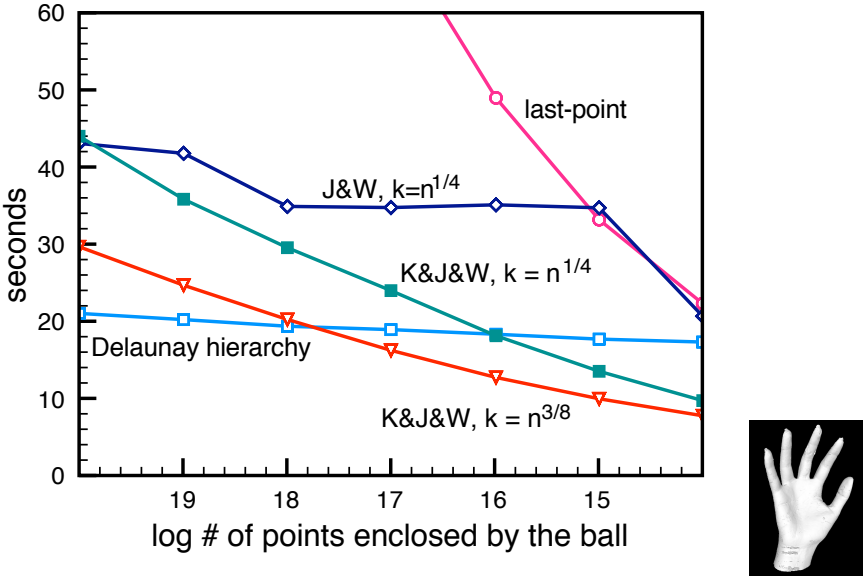
inside the model with the k-last-point strategy after a spatial sorting of the queries. Surprisingly, using a small k slightly improves on $k = 1$ which indicates that even with such a strong coherence, k-last-points strategy is relevant. Table 1 shows the running times on various sets for different values of k, taking $k = 4$ always improves on $k = 1$ and in some cases by a substantial amount.

k	1	2	3	4	5	6	$k = 4$ improves on $k = 1$ by
2D							
uniform square	1.70s	1.65s	1.65s	1.65s	1.66s	1.67s	2%
anisotropic square	1.64s	1.61s	1.60s	1.60s	1.61s	1.62s	1%
ellipse	**3.07s**	**2.73s**	**2.62s**	**2.56s**	**2.54s**	**2.52s**	**17%**
3D							
uniform cube	3.57s	3.45s	3.41s	3.39s	3.40s	3.46s	5%
anisotropic cube	3.45s	3.35s	3.32s	3.31s	3.32s	3.39s	4%
ellipsoid	**6.34s**	**5.71s**	**5.48s**	**5.38s**	**5.34s**	**5.44s**	**15%**
Pooran's Hand	**3.81s**	**3.63s**	**3.58s**	**3.57s**	**3.56s**	**3.63s**	**6%**
Galaad	4.19s	4.08s	4.04s	4.03s	4.07s	4.12s	3%

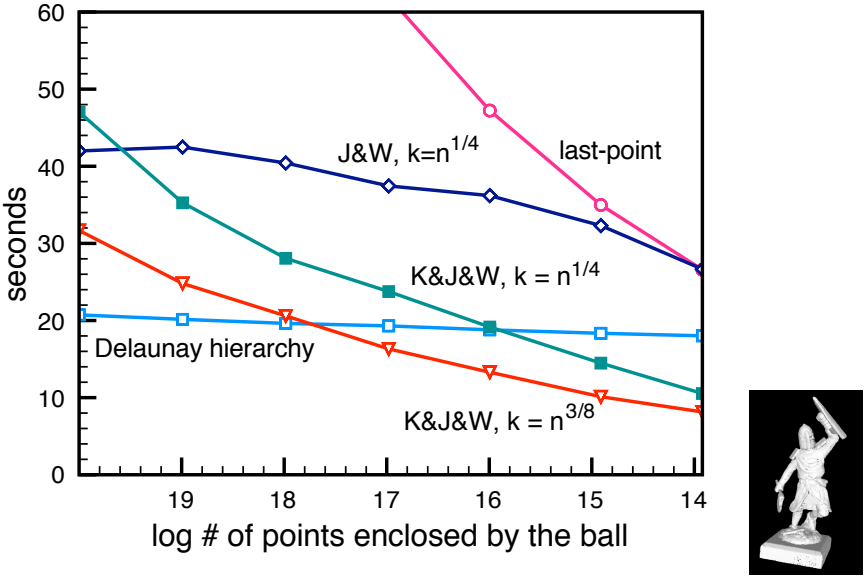
Table 1: **Static point location with space-filling heuristic plus last-k-points strategy.** *Times are in seconds.*

6.3 Self adapting point location

Now, we compare the performance of classical Jump & Walk, Delaunay hierarchy, last-point strategy and Keep, Jump, & Walk. For this purpose we consider the following experiment scenario. Let $\mathcal{M}$ be a scanned model with $1M$ points inside the unit cube, and $\mathcal{B}_r$ be a ball centered at $(0.5, 0.5, 0.5)$ with radius r. Now define S_n as the set of n points on the surface of the model enclosed by $\mathcal{B}_r$. Take values of r such that $n = n_i = 2^i$, for $i = 14, 15, \ldots, 20$. Now form the sequence A_i of $1M$ points taken randomly from S_{n_i} and slightly perturbed (as to avoid arithmetic filter failures). Figure 5 shows the computation times for point location for the different strategies in function of $\log_2 n_i$.



(a)



(b)

Figure 5: **Performance on a dynamic setting.** *Computation times of the various algorithms in function of the number of points on the model enclosed by a ball centered at* $(0.5, 0.5, 0.5)$ *for: (a) Pooran's Hand model; and (b) Galaad model.*

In both models, Keep, Jump, & Walk becomes quickly faster than the classical Jump & Walk when the locality of the queries is stronger. Moreover, Keep, Jump, & Walk becomes faster than the Delaunay hierarchy when the ball encloses less than 64K points. This quantity of points could be multiplied

by approximately 4 if instead of picking $n^{1/4}$ landmarks we pick $n^{3/8}$ landmarks (which is in accord with Conjecture 1, see Section 4). As the number of points enclosed by the ball decreases, Keep, Jump, & Walk improves its performance, and becomes twice faster than the Delaunay hierarchy. The classical Jump & Walk cannot perform better than the Delaunay hierarchy on such big examples.

Finally, complementing the experiments in Section 6.2, consider the sequence A' formed by ordering points in S_{1M} on a space-filling curve. Now A' has a strong spatial coherence. This situation is slightly different from Section 6.2; here the queries are close to the model boundary, while in Section 6.2 they were inside the model. Table 2 shows the performance of the different strategies to locate sequential queries in A', for Pooran's Hand model and Galaad model.

Models	last-point	J&W, $k = 32$	J&W, $k = 101$
Pooran's Hand	3.86s	25.73s	20.01s
Galaad	3.91s	28.41s	21.62s
	Delaunay hierarchy	**32-last-points**	101-last-points
Pooran's Hand	15.14s	**3.49s**	4.30s
Galaad	16.44s	**3.49s**	4.27s

Table 2: **Performance on a static setting.** *Queries close to the model boundary are ordered as to appear in sequence on a space-filling curve. The table shows the computation time to locate the whole sequence of points with different strategies (times are in seconds).*

The improvement k-last-points strategy ($k = 32$) gives with respect to the last-point strategy to locate A' is around 10%, even though A' has a strong spatial coherence.

7 Conclusion

This work discussed how the Distribution Condition and the length of some trees embedded in $\mathbb{R}^d$ can be put together to explain self-adapting variants of well-known algorithms for point location. In the case of query-points with no spatial coherence, this works showed that the constant involved with self-adapting strategies, such as the last-point strategy, is not that bad for evenly distributed points. In particular, Keep, Jump, & Walk has the same computational complexity than Jump & Walk if we use a brute-force nearest neighbor search approach. This work also provides experimental evidences that: (i) realistic data-sets satisfy the Distribution Condition; and (ii) self-adapting variant of the Jump & Walk is rather likely to improve performance, than decrease it, in both static and dynamic settings.

Designing a point location data-structure to retrieve queries in a time logarithmic in the local complexity, without any hypotheses, is a very delicate question [7]. As long as the triangulation satisfies some hypotheses, we showed a simple data-structure which achieves this complexity. The good points here are three: (i) the Delaunay hierarchy is simple and has a good practical behavior [8] (it is currently implemented in CGAL) (ii) the pre-processing and memory complexity are strictly better than previous data-structures [15, 7] (which in

the other hand work for general planar triangulation), (iii) the proposed point location algorithm generalizes for any finite dimension.

Acknowledgments The authors wish to thank Aim@shape for providing the realistic models.

References

[1] Pierre Alliez, Laurent Saboret, and Nader Salman. Point set processing. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009. http://www.cgal.org/Manual/3.5/doc_html/cgal_manual/packages.html#Pkg:PointSetProcessing

[2] Sunil Arya, Theocharis Malamatos, and David M. Mount. A simple entropy-based algorithm for planar point location. *ACM Trans. Algorithms*, 3(2):17, 2007.

[3] D. H. Bailey, J. M. Borwein, and R. E. Crandall. Box integrals. *J. Comput. Appl. Math.*, 206(1):196–208, 2007.

[4] J. Beardwood, J. H. Halton, and J. M. Hammersley. The shortest path through many points. *Math. Proc. Camb. Phil. Soc.*, 55:299–327, 1959.

[5] CGAL Editorial Board. *CGAL User and Reference Manual*, 3.5 edition, 2009. http://www.cgal.org/Manual/3.5/doc_html/cgal_manual/packages.html.

[6] Christophe Delage. Spatial sorting. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009. http://www.cgal.org/Manual/3.5/doc_html/cgal_manual/packages.html#Pkg:SpatialSorting.

[7] Erik D. Demaine, John Iacono, and Stefan Langerman. Proximate point searching. *Comput. Geom. Theory Appl.*, 28(1):29–40, 2004.

[8] Olivier Devillers. The Delaunay hierarchy. *Internat. J. Found. Comput. Sci.*, 13:163–180, 2002.

[9] Olivier Devillers, Sylvain Pion, and Monique Teillaud. Walking in a triangulation. *Internat. J. Found. Comput. Sci.*, 13:181–199, 2002.

[10] Luc Devroye, Christophe Lemaire, and Jean-Michel Moreau. Expected time analysis for Delaunay point location. *Comput. Geom. Theory Appl.*, 29:61–89, 2004.

[11] Luc Devroye, Ernst Peter Mücke, and Binhai Zhu. A note on point location in Delaunay triangulations of random points. *Algorithmica*, 22:477–482, 1998.

[12] J. Gao and J. M. Steele. General spacefilling curve heuristics and limit theory for the traveling salesman problem. *Journal of Complexity*, 10:230–245, 1994.

[13] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press LLC, Boca Raton, FL, 2004. 2nd edition.

[14] John Iacono. Expected asymptotically optimal planar point location. *Comput. Geom. Theory Appl.*, 29(1):19–22, 2004.

[15] John Iacono and Stefan Langerman. Proximate planar point location. In *Proc. 19th Annu. Symp. Comp. Geom.*, pages 220–226, 2003.

[16] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12(1):28–35, 1983.

[17] Ernst P. Mücke, Isaac Saias, and Binhai Zhu. Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations. In *Proc. 12th Annu. Sympos. Comput. Geom.*, pages 274–283, 1996.

[18] Sylvain Pion and Monique Teillaud. 3D triangulations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009. http://www.cgal.org/Manual/3.5/doc_html/cgal_manual/packages.html#Pkg:Triangulation3.

[19] L. K. Platzman and J. J. Bartholdi, III. Spacefilling curves and the planar travelling salesman problem. *J. ACM*, 36(4):719–737, October 1989.

[20] Anderssen R., R. Brent, D. Daley, and P. Moran. Concerning $\int_0^1 \dots \int_0^1 \sqrt{x_1^2 + \dots + x_k^2} dx_1 \dots dx_k$ and a Taylor Series Method. *SIAM J. Appl. Math.*, 30:22–30, 1976.

[21] Luis A. Santaló. *Integral Geometry and Geometric Probability*. Addison-Wesley, 1976.

[22] Steven S. Skiena. *The Algorithm Design Manual*. Springer, 2008.

[23] J. M. Steele. Cost of sequential connection for points in space. *Operations Research Letters*, 8:137–142, 1989.

[24] J. M. Steele and T. L. Snyder. Worst-case growth rates of some classical problems of combinatorial optimization. *SIAM J. Comput.*, 18:278–287, 1989.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399