Sigma-Local Graphs

Prosenjit Bose [1] Sébastien Collette [2] Stefan Langerman [3]
Anil Maheshwari [1] Pat Morin [1] Michiel Smid [1]

[jit,maheshwa,morin,michiel]@scs.carleton.ca
School of Computer Science, Carleton University, Canada

[sebastien.collette,stefan.langerman]@ulb.ac.be
Computer Science Department, Université Libre de Bruxelles, Belgium

Abstract

We introduce and analyze σ-local graphs, based on a definition of locality by Erickson [12]. We present two algorithms to construct such graphs, for any real number $\sigma > 1$ and any set S of n points. These algorithms run in time $O(\sigma^d n + n \log n)$ for sets in $\mathbb{R}^d$ and $O(n \log^3 n \log\log n + k)$ for sets in the plane, where k is the size of the output.

For sets in the plane, algorithms to find the minimum or maximum σ such that the corresponding graph has properties such as connectivity, planarity, and no-isolated vertex are presented, with complexities in $O(n \log^{O(1)} n)$. These algorithms can also be used to efficiently construct the corresponding graphs.

1 Introduction

We consider here *proximity graphs* [15], also called *neighborhood graphs*. Such graphs are defined on a finite set S of points in $\mathbb{R}^d$ (where the dimension d is constant) used as the vertices of the graph and there exists an edge between any two vertices if they are *close* in some sense. The proximity can be measured for instance by the Euclidean distance between these vertices, the distance to other vertices of the graph, or the number of other vertices in a given neighborhood.

[1] Partially supported by NSERC.
[2] Chargé de recherches du F.R.S.-FNRS.
[3] Maître de recherches du F.R.S.-FNRS.

A graph is σ-local [12] if for every edge, there exists a ball around each of its endpoints not containing any other vertex. If the distance between the endpoints of an edge is denoted by d, the balls must have a radius of at least d/σ. We introduce a new parameterized family of proximity graphs: a *σ-local graph* on a given set of points is the maximal graph fulfilling the σ-local property for all its edges. We define this formally in the next section.

Proximity graphs are well studied; a survey of Jaromczyk and Toussaint [15] discusses many of them, such as relative neighborhood graphs [15, 2], Gabriel graphs [13], β-skeletons [17], and rectangular influence graphs [14]. The Θ-graphs [16, 24] and γ-neighborhood graphs [23] are other examples. Among these, some definitions encompass *families* of proximity graphs. For instance, the β-skeletons form a parameterized family, in which the proximity definition depends on a parameter β.

Previous work on proximity graphs traditionally consisted in the introduction of one or more graph families, followed by different contributions analyzing their properties and applications. For instance, in [18] the properties of the Gabriel graphs were used to analyze geographical data sets; in [7], the authors analyze the spanning ratio of some proximity graphs. The natural opposite approach does not seem to have been widely investigated: to try to find a practical method which, given desired graph properties, constructs a corresponding proximity graph. A first step towards the definition of customizable proximity graphs given a list of desirable properties was proposed in [9].

The aim of this work is to find a family of graphs which is easy to define, and where the choice of the parameter σ leads to different interesting properties, such as planarity, connectivity, *etc.* σ-local graphs fulfill these requirements nicely for different reasons: first their definition seems natural, because when $\sigma = 1$, σ-local graphs are equivalent to nearest neighbor graphs [11]; we will prove that finding the extremal values of σ for which the aforementioned properties are present can be achieved in $O(n \log^{O(1)} n)$ time [4]; and moreover, we will show that they can be constructed efficiently.

Related Works

Originally, Erickson [12] introduced this definition of σ-locality as a realistic input model for non-convex polyhedra. In general, collision detection for non-convex polyhedra, as well as Boolean operations on these polyhedra are costly operations. By enforcing the input to be σ-local, Erickson showed that Boolean

[4] The exponent is a small constant. For instance, checking if a graph is connected can be achieved in $O(n \log^7 n)$ time.

combinations of two (σ-local) polyhedra defined by n vertices in $\mathbb{R}^d$ can be performed in $O(n \log n)$ time.

We provide the first approaches to find the extremal values of σ ensuring some properties, which can be used to simplify the input. Having a lower value for σ is important in practice, because the complexities in Erickson's work are actually heavily influenced by σ (which, in [12], is used as a constant, and thus hidden in the $O()$-notation).

This work is of course strongly related to the tremendous literature on proximity graphs. And, as mentioned above, it is related in particular to the approach initiated with empty region graphs [9], which consists in defining the proximity constraint based on properties we want for the resulting graph. The main difference between empty region graphs and σ-local graphs is that the formers ensure that a proximity constraint will guarantee a property for every set of vertices; while here we propose algorithms to determine the extremal value of σ for the given property to be satisfied on a *given* set of vertices.

If this work might resemble previous works on geometric graphs, we think that the motivation here is different. The abundance of knowledge on proximity graphs has a drawback: if, for some application, one wants to find a family of graphs with a given list of required properties, all the literature must be reviewed to check what property is guaranteed by what graph family. Our work aims at simplifying this process.

Outline

In Section 2 we formally define σ-local graphs and give examples. Section 3 presents algorithms to construct σ-local graphs for a given σ. Different approaches are proposed depending on the value of σ and on the dimension.

Section 4 presents different algorithms to test if, given a set of points and a parameter σ, the corresponding σ-local graph contains isolated vertices, if it is connected, and if it is plane when the edges are embedded as straight line segments. We provide algorithms which actually find the *optimal* σ, i.e., the minimum or maximum value ensuring these properties.

2 Definitions

The following definition was proposed by Jeff Erickson [12]. Let $N(v)$ denote the set of neighbors of a vertex v in the graph G. We denote the Euclidean distance between two points p and q by $|pq|$.

Definition 1 *A geometric graph $G(S, E)$ is* σ-local *if its local stretch $\sigma(G)$ is less than a fixed parameter σ and its global stretch $\Sigma(G)$ is bounded from above by a fixed polynomial in the number of vertices, where*

$$\sigma(v) = \frac{\max\limits_{u \in N(v)} |uv|}{\min\limits_{u \in S \setminus \{v\}} |uv|}$$

$$\sigma(G) = \max_{v \in S} \sigma(v)$$

$$\Sigma(G) = \frac{\max\limits_{uv \in E} |uv|}{\min\limits_{uv \in E} |uv|}$$

Intuitively, this means that in every σ-local graph, there exists an edge between a pair of vertices (p, q) only if the two balls centered at p and q with radius $|pq|/\sigma$ are empty, and the maximum ratio between the length of any pair of edges is bounded. This is equivalent to requiring that all edges pq satisfy $|pq| \leq \sigma \cdot \min(W_p, W_q)$, where the weight W_p of a point p is the distance to its nearest neighbor; see Figure 1.

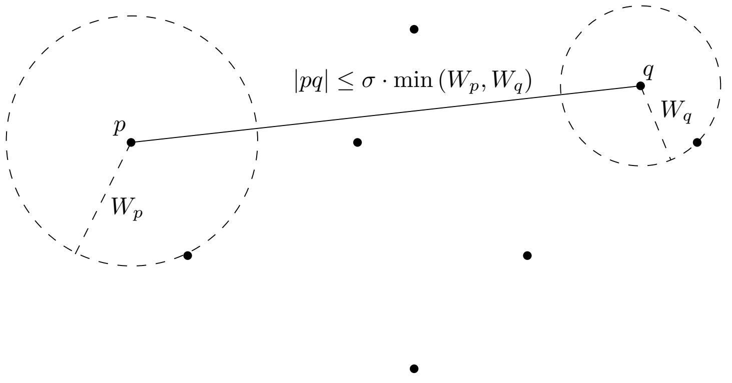

Fig. 1. Restriction in σ-local graphs.

Erickson used this definition of σ-locality as a realistic input model. The polynomial bound on $\Sigma(S)$ is needed for the analysis of complexities in [12], we will not consider it here.

We propose a new family of proximity graphs using the same definition of locality:

Definition 2 *Let S be a set of n points in $\mathbb{R}^d$, and let $\sigma \geq 1$ be a real number. The* σ-local graph *of S, denoted by $G_\sigma(S)$, is defined to be the graph with vertex set S and edge set E, where (p, q) is in E if and only if*

(1) the ball with center p and radius $|pq|/\sigma$ does not contain any point of $S \setminus \{p, q\}$ in its interior, and

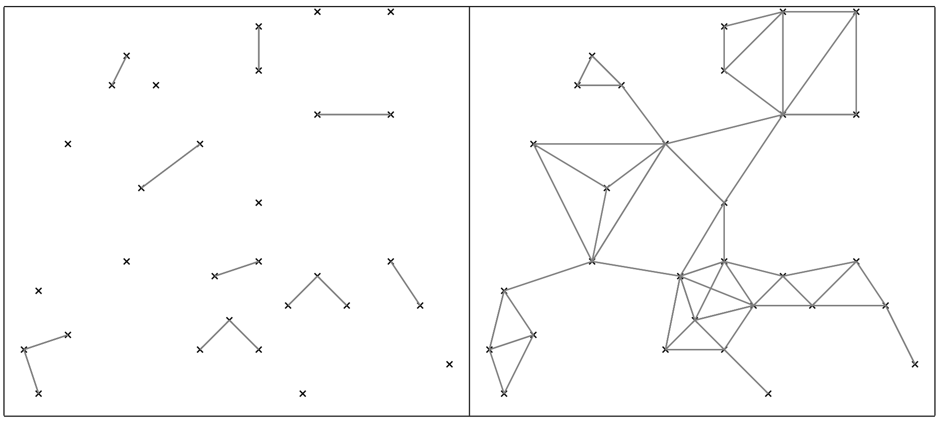
Fig. 2. σ-local graphs for parameters 1 and 2.

(2) the ball with center q and radius $|pq|/\sigma$ does not contain any point of $S \setminus \{p, q\}$ in its interior.

This corresponds to a proximity graph using the pair of balls centered at p and q with radius $|pq|/\sigma$ as an exclusion region [9] for the edge pq: there exists an edge if and only if the exclusion region is empty. Some examples of σ-local graphs are given by Figure 2.

3 Construction of $G_\sigma(S)$

In this section we show how to efficiently construct the corresponding σ-local graph given a set of vertices and a value σ.

We can easily construct σ-local graphs as follows: using Vaidya's algorithm [22], we compute, for each point p in S, its weight W_p (the distance to its nearest neighbor). This can be done in $O(n \log n)$ time in any constant dimension. Then, we can test all possible edges in $O(n^2)$ time, and keep only the ones satisfying the σ-local condition. In the remainder of this section, we propose two sub-quadratic algorithms: one whose complexity depends on σ, and a second whose complexity depends on the size of the graph.

The main idea is to avoid to test all edges. In the first algorithm, we use the well-separated pair decomposition to rapidly find candidate edges, and in the second one we combine circular range queries and an ad-hoc data-structure to get an efficient algorithm.

Note that we do not consider the case where $\sigma = 1$, as this has been studied before in the context of nearest neighbor graphs [11].

3.1 Fixed σ

The first algorithm we present is valid for any real number $\sigma > 1$ and in any (constant) dimension d. However the running time is interesting only if σ^d is much smaller than n. We propose a more efficient solution for the other case in the next subsection, but only for sets of points in the plane.

Given a separation ratio α, two point sets are well-separated if they can be enclosed in two spheres of radius ρ such that the distance between the two spheres is at least $\rho\alpha$. Given a set S of n points in $\mathbb{R}^d$, a well-separated pair decomposition [8] is a set of $m = O(\alpha^d n)$ pairs of sets of points $\{A_i, B_i\}$ such that:

- For all p and q in S, there exists a unique value of $1 \leq i \leq m$ for which $p \in A_i$ and $q \in B_i$.
- For $1 \leq i \leq m$, A_i and B_i are well-separated (and in particular, A_i and B_i are disjoint sets for every i).

Let $\sigma > 1$ and consider a well-separated pair decomposition $\{A_i, B_i\}$ with separation ratio 2σ. We obtain the following lemma:

Lemma 3 *Let (p, q) be an edge in $G_\sigma(S)$, and let i be the index such that $p \in A_i$ and $q \in B_i$. Then both A_i and B_i are singleton sets*[5] *.*

PROOF. Assume that A_i contains a point r of S with $r \neq p$. Since the sets A_i and B_i are well-separated, there exist balls C and C' having the same radius ρ, such that $A_i \subseteq C$, $B_i \subseteq C'$, and the distance between C and C' is at least $2\sigma\rho$. Since p and r are both contained in C, we have $|pr| \leq 2\rho$. On the other hand, since $p \in C$ and $q \in C'$, we have $|pq| \geq 2\sigma\rho$. By combining these inequalities, it follows that $|pr| \leq |pq|/\sigma$, contradicting the fact that (p, q) is an edge in the σ-local graph $G_\sigma(S)$. □

Theorem 4 *The σ-local graph $G_\sigma(S)$ for a point set S in $\mathbb{R}^d$ containing n points and a given σ can be constructed in $O(\sigma^d n + n \log n)$ time.*

PROOF. As described above, using Vaidya's algorithm [22], we compute, for each point p in S, its weight W_p (the distance to its nearest neighbor). This can be done in $O(n \log n)$ time in any constant dimension.

Using Callahan and Kosaraju's algorithm [8], we compute a well-separated pair decomposition $\{A_i, B_i\}$, $1 \leq i \leq m$, for S, with separation ratio 2σ,

[5] Note that the converse of this lemma is, in general, *not* true.

where $m = O(\sigma^d n)$. This is achieved in $O(n \log n + \sigma^d n)$ time.

Then, we compute the index set I consisting of all indices i such that both A_i and B_i are singleton sets, in $O(\sigma^d n)$ time.

We initialize an empty edge set E. For each $i \in I$, let p_i and q_i be the points of S such that $A_i = \{p_i\}$ and $B_i = \{q_i\}$. We add the edge (p_i, q_i) to E if and only if both W_{p_i} and W_{q_i} are at most $|p_i q_i|/\sigma$. This step takes $O(\sigma^d n)$ time.

The algorithm finishes by returning the graph $G = (S, E)$, which, by Lemma 3 is the σ-local graph $G_\sigma(S)$. $\square$

This also shows that the σ-local graph of S contains $O(\sigma^d n)$ edges. This upper bound is only meaningful if $\sigma^d = o(n)$.

3.2 Large Fixed σ and $d = 2$

We present here an algorithm which is efficient for point sets in the plane.

Theorem 5 *The σ-local graph $G_\sigma(S)$ for a point set S in $\mathbb{R}^2$ containing n points and a given σ can be constructed in $O(n \log^3 n \log\log n + k)$ time, where k is the number of edges in the resulting graph.*

PROOF. For each point $p \in S$, we want to compute all $q \in S$, such that

(1) $\sigma W_p \geq |pq|$, and
(2) $\sigma W_q \geq |pq|$.

That is, for each point p we want to find all points q such that the balls of radius $|pq|/\sigma$ around both p and q are empty. In this way, we obtain all edges (p, q) that are incident on p. Let $p = (p_1, p_2)$ and $q = (q_1, q_2)$. We denote by $S_{p,<}$ and $S_{p,>}$ the sets of points $\{q \in S | W_q \leq W_p\}$ and $\{q \in S | W_q > W_p\}$, respectively. We have to analyze two subproblems:

(1) Given p, find the points q in $S_{p,>}$ such that $\sigma W_p \geq |pq|$.
(2) Given p, find the points q in $S_{p,<}$ such that $\sigma W_q \geq |pq|$.

However, since the graph is undirected we do not have to consider both cases. For every edge pq in $G_\sigma(S)$, $W_p \geq W_q$ or $W_p \leq W_q$. Thus, an edge will be constructed when we consider its endpoint with lower weight and we only have to solve subproblem (1).

We store the points of S at the leaves of a balanced binary search tree, sorted by their weights. At each node u of this tree, we store a secondary data structure as outlined below. The depth of the tree is $O(\log n)$, and at each level, the canonical sets represent exactly n vertices divided into disjoint sets.

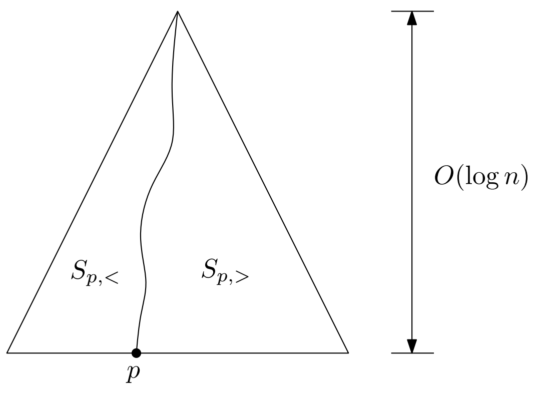

Fig. 3. Binary tree associated to the point set.

Given a query point p, we want to find all $q \in S_{p,>}$ such that $\sigma W_p \geq |pq|$. We search in the tree for p. The search path partitions the set of all q with $W_p \leq W_q$ into $O(\log n)$ canonical nodes. (These are the right children of nodes on the path in which the path moves to the left child, see Figure 3.)

For each canonical node u, we want to find all points q in the subtree of u for which $\sigma W_p \geq |pq|$. These are all points q that are in the circle with center p and radius σW_p. So we store at u a data structure that supports these queries. Thus, given the data structure, the time to construct the σ-local graph is $O(n \log n)$ times the time for answering circular range reporting queries.

Aggarwal, Hansen, and Leighton [3] provide a data-structure that supports circular range reporting queries in $O(\log n + k)$ time where k is the number of items reported. The structure can be built in $O(n \log^2 n \log\log n)$ time and uses $O(n \log n)$ space.

Since the primary tree has $O(\log n)$ levels, and since the secondary data structures are disjoint and each level encompasses a linear number of nodes, the total time complexity to construct $G_\sigma(S)$ is thus $O(n \log^3 n \log\log n + k)$. $\square$

4 Testing Properties of $G_\sigma(S)$

In this section, we test properties of σ-local graphs for a given value of σ and point set S. We also propose algorithms to find the threshold value of that parameter to satisfy the given property on that particular set.

4.1 No Isolated Vertex

A natural question when we want to analyze a graph is to determine if vertices are always part of connected components, or if they are isolated. Given a set of vertices, another view of the problem is to find the smallest σ such that the corresponding graph contains no isolated vertex.

Theorem 6 *Given a set S of n points in $\mathbb{R}^2$, the σ-local graph $G_\sigma(S)$ with lowest σ such that $G_\sigma(S)$ contains no isolated vertex can be constructed in $O(n \log^5 n)$ time.*

PROOF. To find the smallest value of σ, we look at all the vertices and determine which value of σ connects them to another vertex in the graph. As we want to check if some vertex is isolated, we must consider both ends of each edge and not only the endpoint with lowest weight as we did in Theorem 5.

We describe the solution in three phases. First we show the preprocessing phase, in which we construct a data structure which will let us check if a point is isolated for a given σ. Second, we give a parallel algorithm using that data structure and answering queries for a fixed σ, and third we show how to derive an algorithm finding the optimal σ.

Preprocessing phase, in $O(n \log^2 n)$ time. As in Theorem 5, we use a balanced binary search tree T to store the vertices according to their weights. We associate to each internal node the Voronoi diagram of the set it represents. The depth of the tree is $O(\log n)$, and at each level, the canonical sets represent exactly n vertices divided in disjoint sets.

As the Voronoi diagram of n points can be computed in $O(n \log n)$ time, the total complexity to construct the n Voronoi diagrams associated to the internal nodes is thus $O(n \log^2 n)$: for each level in the tree, the complexity is bounded by $O(n \log n)$, and there are $O(\log n)$ levels.

Using the data structure created in this preprocessing phase, we can easily check if each point p connects to a point with higher weight: for the vertices in $S_{p,>}$, the vertex q leading to the lowest σ is the closest from p, because σ only depends on the distance $|pq|$ and W_p. Given a Voronoi diagram of the point set $S_{p,>}$, finding the point q leading to lowest σ is a point location problem: by definition it is the point whose Voronoi cell contains p.

For every point p, we follow the path from the root to the leaf of the tree representing that point. The union of all the sets corresponding to the right children of nodes on that path constitutes the set $S_{p,>}$. Thus we need to perform a query in $O(\log n)$ Voronoi diagrams, and each query costs $O(\log n)$

time. This gives us $O(\log n)$ candidates for the closest point, and a linear search gives us the minimum σ.

So at the end of this phase, we have, for every point p, the minimum value of σ for which it would connect to a point in $S_{p,>}$.

Parallel algorithm for a fixed σ, in $O(n \log^3 n)$ time. To check if there is no isolated vertex when the graph is built using a given σ, we start by the preprocessing phase. For the vertices p that are not connected to a point in $S_{p,>}$ using that value of σ, we still have to check that they are not connected to a point in $S_{p,<}$.

Therefore, we use the following approach: we look for a vertex $q \in S_{p,<}$ such that the edge (p, q) is compatible with the value of σ. The vertex p is connected to the vertex q if and only if

$$\frac{|pq|}{W_q} \leq \sigma$$

We lift the problem to $\mathbb{R}^3$ by mapping each vertex $q \in S_{p,<}$ to a plane P_q given by

$$P_q \equiv 2\, q_1\, x + 2\, q_2\, y + z = q_1^2 + q_2^2 - \sigma^2\, W_q^2$$

where q_1 and q_2 are respectively the first and the second coordinate of the point q. Thus the edge will appear for the value σ if the point $(p_1, p_2, -p_1^2 - p_2^2)$ is below P_q:

$$\begin{aligned}\frac{|pq|}{W_q} \leq \sigma &\iff 2\, q_1 p_1 + 2\, q_2 p_2 + (-p_1^2 - p_2^2) \geq q_1^2 + q_2^2 - \sigma^2\, W_q^2 \\ &\iff (p_1, p_2, -p_1^2 - p_2^2) \text{ is below } P_q\end{aligned}$$

By computing the upper envelope of all the planes corresponding to the set $S_{p,<}$, we can check if p is connected to any point of that set. We proceed as follows:

(1) For each node v of T in parallel, we use the parallel 3D convex hull algorithm of Amato and Chow [4], to compute the upper envelope of the planes defined by points in the subtree rooted at v. This can be achieved using $O(n \log n)$ processors in $O(\log^2 n)$ time.
(2) For each node v of T in parallel, we use the parallel algorithm of Reif and Sen [21] to build a point location structure for the upper envelope computed in Step 1 - achieved with $O(n \log n)$ processors in $O(\log n)$ time.
(3) For each point p in parallel, we locate p in each of its $O(\log n)$ relevant point location structures to determine whether, for the current value of

σ, the point p is connected to a point of lower weight. We use $O(n \log n)$ processors, and $O(\log n)$ time.

(4) For each point p, it is not isolated if it is connected to a point of lower weight, or if σ is greater than the value for p computed in the preprocessing phase. Using $O(n)$ processors, it takes $O(\log n)$ time.

In conclusion, we have a parallel algorithm running in $O(n \log^3 n)$ total time which checks if there is no isolated vertex for a fixed σ.

Note that if we are just interested in a sequential algorithm for a fixed σ, we can easily derive a $O(n \log^2 n)$ time algorithm using the same approach as for the parallel algorithm, as we can use an optimal 3D convex hull algorithm, which does not need to be parallel.

Algorithm finding the lowest σ, in $O(n \log^5 n)$ time. To find the lowest σ, we use the parametric search technique of Megiddo [19] which transforms a decision algorithm into an optimization algorithm efficiently.

We denote by T_s the running time of a sequential decision algorithm A. If A_p is an equivalent parallel algorithm running in T_p parallel steps using P processors, Megiddo proved that the total running time of the optimization algorithm using his technique is $O(PT_p + T_pT_s \log P)$. By applying this to our decision algorithm for a fixed σ described above, we get, if we run it on $P = n \log n$ processors, that $O(n \log^3 n + \log^2 n \cdot n \log^2 n \cdot \log(n \log n))$, which evaluates to $O(n \log^5 n)$ total time. □

4.2 Connectedness

Next we consider the problem of finding the minimum value of σ such that $G_\sigma(S)$ is connected. For this, we apply ideas used in Boruvka's algorithm [6, 20] for finding the minimum spanning tree.

Theorem 7 *Given a set S of n points in $\mathbb{R}^2$, the connected σ-local graph with minimum σ can be constructed in $O(n \log^7 n)$ time.*

PROOF. The algorithm iteratively groups vertices into components, as σ grows, up to the point where there is a single component. We start with components of size one, i.e., every vertex is isolated. We put every vertex in the leaves of a binary tree, sorted by component number; the internal nodes of that tree represents (sub-)sets of components. Each internal node represents a set of vertices for which we create a secondary data structure as described in Subsection 4.1, with Voronoi diagrams and support for circular range queries, to be able to check if any vertex in any of the components represented by one internal node is isolated.

In the binary tree, we can easily find sets covering every vertex in all components but one: from the root of the binary tree, there is a left-most and a right-most path (see Figure 4), leading respectively to the first and the last leaf of a component. The left (right resp.) children of the internal nodes on the left- (right- resp.) most path represent covering sets for all the other components.

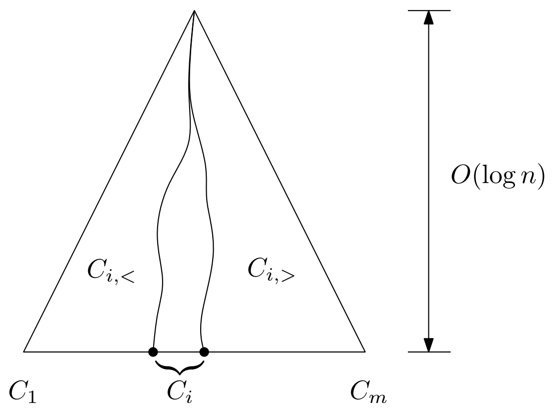

Fig. 4. Binary tree associated to the components.

Then we find the smallest σ which connects one component to any other component by running the no-isolated vertex algorithm for every point in that component on the points in the other components.

This reduces the number of components by a factor of (at least) two, as every time we scan the whole set of points, every component gets connected to one other. We continue up to the point where there is only one component; we can then return the lowest σ.

As the number of components is reduced each time by a factor of two, we repeat the no-isolated vertex algorithm $O(\log n)$ times, on $O(\log n)$ sets covering every component but the current one.

The time to initially construct and maintain the structure is dominated by the search of the optimum value; the complexity is thus $O(n \log^7 n)$. $\square$

4.3 Planar Embedding

The last property we analyze is whether the σ-local graph obtained is a planar embedding[6]. The following algorithm allows us to find the largest value of σ

[6] We only consider straight-line drawings, i.e., an edge is a line segment between two vertices.

for which this is the case, and also constructs the corresponding graph in the meantime.

Theorem 8 *Given a set S of n points in $\mathbb{R}^2$, the plane σ-local graph with maximum σ can be constructed in $O(n \log^{O(1)} n)$ time.*

To prove this Theorem, we will first need a technical lemma, which will be used later to construct graphs with more than a given number of edges:

Lemma 9 *Given a point set S in $\mathbb{R}^2$, we can find the smallest real number σ such that $G_\sigma(S)$ has $n \log^3 n \log\log n$ edges in $O(n \log^{O(1)} n)$ time.*

PROOF. First we consider the following decision problem: Given the set S, and given σ, decide if $G_\sigma(S)$ contains at least K edges. For this, we could simply execute the algorithm used for Theorem 5 and stop as soon as we have enough edges. But, since we will later use the parametric search technique [19], we want to make sure that the algorithm can be run in the PRAM model, so we propose the following alternative approach.

The circular range reporting query in two dimensions corresponds to a half-space query in 3D [1]. Let S' be the 3D set corresponding to S; the procedure described hereunder allows us to perform half-space queries on S'.

Let T be a balanced tree that stores the points of S' at its leaves, in an arbitrary order. With each node u of T, we store the Dobkin-Kirkpatrick hierarchy [10] of the convex hull of all points in the subtree of u.

To answer a query, we are given a plane P in $\mathbb{R}^3$ and want all points of S' that are below P. We start at the root of T. Using the Dobkin-Kirkpatrick hierarchy, we can decide in $O(\log n)$ time if the convex polyhedron stored at the root

(1) is completely above P: in this case, the query algorithm terminates,
(2) is completely below P: in this case, we report all points stored in the tree and terminate,
(3) intersects P: in this case, we recursively query both subtrees of the root.

Let k be the number of points that are below P. The number of nodes of T that are visited by the query algorithm is $O(k \log n)$. At each node, the algorithm spends $O(\log n)$ time. Of course, the algorithm can terminate as soon as K edges have been reported.

These queries can be solved in parallel: in case 3 of the query algorithm, we use two processors. In other words, each time we branch in the tree, we use an additional processor [7] .

The decision problem (does $G_\sigma(S)$ have at least K edges?) can be solved sequentially, in $O((n+K)\log^2 n)$ time; it can be solved in parallel in $O(\log^{O(1)} n)$ time, using $n + 2K$ processors and only algebraic predicates.

This immediately allows us to use the parametric search technique of Megiddo [19] to find the smallest σ such that $G_\sigma(S)$ contains at least $K = n\log^3 n \log\log n$ edges in time $O(n\log^{O(1)} n)$. □

The choice of K in the previous proof might sound arbitrary, but is based on a precise evaluation of the log factors in the complexity of the algorithm: the algorithm runs in $O(n\log^3 n \log\log n + 2K)$ time. However, as the exact complexity is not needed in the remainder of the paper, we left the details.

PROOF. [Theorem 8] Using Lemma 9, we can construct in $O(n\log^{O(1)} n)$ time a graph containing more than $3n - 6$ edges. We sort these edges by increasing values of σ in $O(n\log^{O(1)} n)$ time [8] and keep the $3n - 6$ first ones. This gives us an upper bound on the value of σ, as a planar graph has $3n - 6$ edges or less.

Given a set of n line segments in $\mathbb{R}^2$, the algorithm proposed by Bentley and Ottmann [5] returns all pairs of segments intersecting each other in $O((n + k)\log n)$ time, where k is the number of intersections in the set. The same algorithm can be used to check if at least one segment intersects with any other, by stopping as soon as it finds one intersection; this can be achieved in $O(n\log n)$ time.

We do a binary search on the size s of the set of edges (initialized to $3n - 6$), using Bentley-Ottmann to check if the first s edges in sorted order intersect or not. This gives a time complexity of $O(n\log^2 n)$ to determine the maximum value of σ corresponding to a planar σ-local graph. □

[7] We might actually create a branch on a new processor that will not generate an answer and will terminate after one level; thus, to find one answer, we need two processors, hence the 2K appearing later.
[8] In other words, we sort the edges by their order of appearance as the value of σ grows.

5 Future Work

We gave algorithms to construct and check different properties of σ-local graphs. Other properties that we would like to study include checking whether the resulting graph is a triangulation (or contains a triangulation as subgraph) for a given σ, and what is the minimum value of σ for this to be true.

Generalization to the k^{th} order, where there is an edge between two vertices if there are less than k other points of the set in each influence ball seems feasible by determining the k^{th} nearest neighbor of each vertex and using the distance to that point as weight. Note however that some care must be taken with our different proofs: for instance Lemma 3 does not hold anymore, as A_i and B_i could contain k items, and every k^2 pairs of items should be considered.

Acknowledgments

This work was initiated during the Carleton-Eindhoven Workshop on Computational Geometry 2005 organized by Prosenjit Bose and Mark de Berg. We would like to thank all participants who contributed in one way or another to the completion of this work.

We also thank the anonymous referees whose comments helped us to improve the presentation of the results.

References

[1] P. Agarwal. Range searching. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 36, pages 809–837. CRC Press LLC, Boca Raton, FL, 2004.

[2] P. Agarwal and J. Matoušek. Relative neighborhood graphs in three dimensions. *Computational Geometry: Theory and Applications*, 2:1–14, 1992.

[3] A. Aggarwal, M. Hansen, and T. Leighton. Solving query-retrieval problems by compacting Voronoi diagrams. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 331–340, New York, NY, USA, 1990. ACM Press.

[4] N. M. Amato and F. P. Preparata. A time-optimal parallel algorithm for three-dimensional convex hulls. *Algorithmica*, 14(2):169–182, 1995.

[5] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28(9):643–647, 1979.

[6] O. Boruvka. O jistem problemu minimalnim. *Prace Moravske Prirodovedecke Spolecnosti 3*, pages 37–58, 1926.

[7] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick. On the spanning ratio of Gabriel graphs and β-skeleton. 20(2):412–427, 2006.

[8] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

[9] J. Cardinal, S. Collette, and S. Langerman. Region counting graphs. In *Proceedings of the European Workshop on Computational Geometry (EWCG05)*, 2005.

[10] D. Dobkin and D. Kirkpatrick. Fast detection of poyhedral intersection. *Theoretical Computer Science*, 27:241–253, 1983.

[11] D. Eppstein, M. Paterson, and F. Yao. On nearest-neighbor graphs. *Discrete and Computational Geometry*, 17:263–282, 1997.

[12] J. Erickson. Local polyhedra and geometric graphs. *Computational Geometry: Theory and Applications*, 31(1-2):101–125, 2005.

[13] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[14] M. Ichino and J. Sklansky. The relative neighborhood graph for mixed feature variables. *Pattern Recognition*, 18:161–167, 1985.

[15] J. Jaromczyk and G. Toussaint. Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80(9):1502–1571, 1992.

[16] J. Keil and C. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete and Computational Geometry*, 7(1):13–28, 1992.

[17] D. Kirkpatrick and J. Radke. A framework for computational morphology. *Computational Geometry*, pages 217–248, 1985.

[18] D. Matula and R. Sokal. Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geographical Analysis*, 12:205–222, 1980.

[19] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, 1983.

[20] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[21] J. H. Reif and S. Sen. Optimal parallel randomized algorithms for three-dimensional convex hulls and related problems. *SIAM Journal on Computing*, 21(3):466–485, 1992.

[22] P. M. Vaidya. An $O(n \log n)$ algorithm for the all-nearest neighbors problem. *Discrete & Computational Geometry*, 4:101–115, 1989.

[23] R. Veltkamp. The γ-neighbourhood graph. *Computational Geometry: Theory and Applications*, 1:227–246, 1992.

[24] A. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.