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 [9]. 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. In $\mathbb{R}^2$, we also present a preprocessing method to find the graph corresponding to any σ in linear time using $O(n \log^{O(1)} n)$ space and preprocessing time.

Algorithms to find the minimum or maximum σ such that the corresponding graph has properties such as connectivity, completeness, 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* [12], also called *neighborhood graphs*. Such graphs are defined on a finite set S of points in the plane 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. Proximity graphs are well-studied; a survey of Jaromczyk and Toussaint [12] discusses many of

[1] Partially supported by NSERC.
[2] Aspirant du F.N.R.S.
[3] Chercheur qualifié du F.N.R.S.

them, such as relative neighborhood graphs [12, 2], Gabriel graphs [10], β-skeletons [14], and rectangular influence graphs [11]. The Θ-graphs [13, 20], empty region graphs [7] and γ-neighborhood graphs [19] are other examples.

Among these, some definitions encompass *families* of proximity graphs. For instance, the β-skeletons and the empty region graphs are parameterized families, in which the proximity definition depends on a constant β or on a geometric shape respectively. Such families of graphs have proven useful because the parameters for which some properties are ensured have been characterized [7, 14].

A graph is σ-local [9] 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. Erickson [9] used this definition of σ-locality as a realistic input model for nonconvex polyhedra, and showed that Boolean combinations of two σ-local polyhedra defined by n vertices in $\mathbb{R}^d$ can be performed in $O(n \log n)$ time.

Related to the work of Erickson, we introduce a new family of proximity graphs. A *σ-local graph* is the maximal graph fulfilling the σ-local property for all its edges. In other words, σ-local graphs are proximity graphs whose proximity constraint is that two vertices are close if the edge between them does not violate the definition of locality. We define this formally in the next section.

We present algorithms to construct σ-local graphs for a given set of points. Instead of characterizing the values of σ ensuring the existence of a given property for every graph, we propose to determine the minimum or maximum value of σ for the given property to be satisfied on a set of points.

Further to Erickson's work, the study of σ-local graphs might prove useful for the analysis of all kinds of objects defined in that realistic input model. Another interesting aspect is that we give algorithms to find extremal graphs with given properties. Previous work on proximity graphs consisted in the introduction of one or more graph families, followed by different contributions analyzing their properties and applications. A first step towards the definition of customizable proximity graphs given a list of desirable properties was proposed in [7]. This work extends that study: given a property, we adapt the proximity constraint (i.e. the value of σ), to get a graph with that property.

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. We also present a data-structure which, given σ as a query, returns the corresponding graph efficiently. Section 4 presents different algorithms to test if, given a set of points and a value of σ, the corresponding σ-local graph is complete, if it contains isolated vertices, if it is connected, and if it is plane when the edges are embedded as straight line segments.

2 Definitions

The following definition was proposed by Jeff Erickson [9]. Let $N(v)$ denote the set of neighbors of a vertex v in the graph G.

Definition 1 *A geometric graph $G(S, E)$ is* σ-local *if its local stretch $\sigma(G)$ is less than a fixed constant 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 every edge 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.

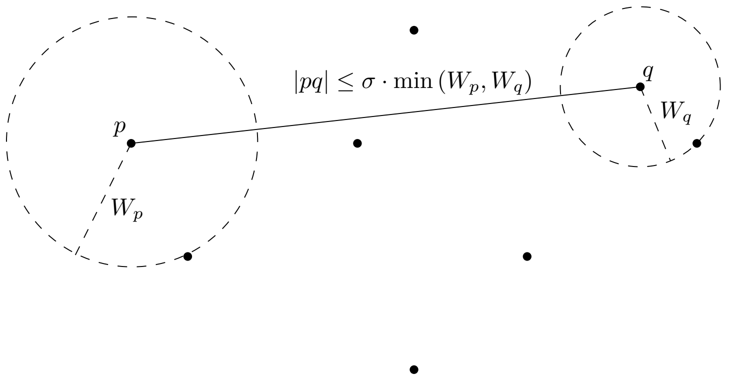

Fig. 1. Restriction in σ-local graphs.

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

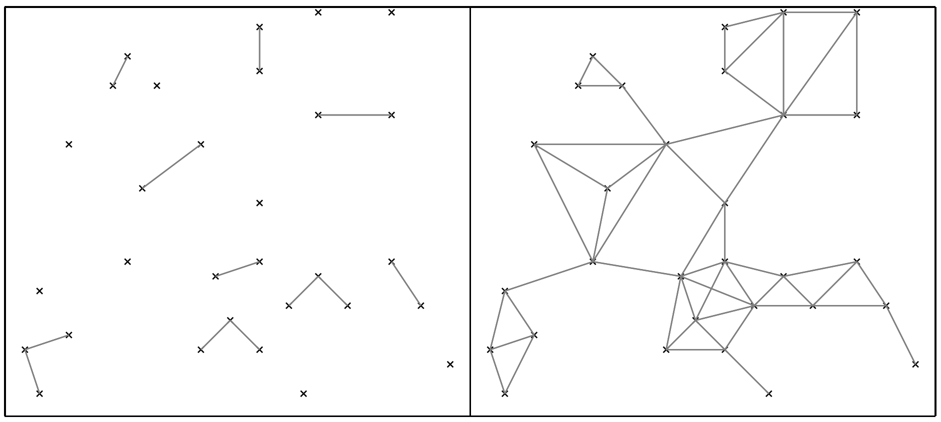

Fig. 2. σ-local graphs for parameters 1 and 2.

Definition 2 *Let S be a set of n points in $\mathbb{R}^d$, and let $\sigma > 1$ 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 $S \setminus \{p\}$ in its interior, and
(2) the ball with center q and radius $|pq|/\sigma$ does not contain any point of $S \setminus \{q\}$ 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 [7] for the edge pq: there exists an edge if and only if the exclusion region is empty.

3 Construction of $G_\sigma(S)$

In this section we show how to efficiently construct the σ-local graph given a set of vertices and a value σ. We present two algorithms: one whose complexity depends on σ, and a second whose complexity depends on the size of the graph.

3.1 Fixed σ

The first algorithm we present is valid for any real number $\sigma > 1$ and in any (constant) dimension d. However if σ^d has the same order of magnitude as n, the running time for the two-dimensional case is sub-optimal, and we propose a more efficient solution for that case in the next subsection.

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 them is at least $\rho\alpha$. Given a set S of n points in $\mathbb{R}^d$, a well-separated pair decomposition [6] 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 $1 \le i \le m$ for which $p \in A_i$ and $q \in B_i$.
- For $1 \le i \le m$, 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.*

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 $|pr| \le 2\rho$. On the other hand, since $p \in C$ and $q \in C'$, we have $|pq| \ge 2\sigma\rho$. By combining these inequalities, it follows that $|pr| \le |pq|/\sigma$, contradicting the fact that (p, q) is an edge in the σ-local graph $G_\sigma(S)$. $\square$

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

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. Using Vaidya's algorithm [18], 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 [6], we compute a well-separated pair decomposition $\{A_i, B_i\}$, $1 \le i \le m$, for S, with separation ratio 2σ, 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

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) $\sigma W_q \geq |pq|$.

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 can rewrite the two conditions as:

(1) $(p_1 - q_1)^2 + (p_2 - q_2)^2 \leq (\sigma W_p)^2$, and
(2) $(p_1 - q_1)^2 + (p_2 - q_2)^2 \leq (\sigma W_q)^2$.

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 $(p_1-q_1)^2+(p_2-q_2)^2 \leq (\sigma W_p)^2$.
(2) Given p, find the points q in $S_{p,<}$ such that $(p_1-q_1)^2+(p_2-q_2)^2 \leq (\sigma W_q)^2$.

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.

To solve the above subproblem, we store the points of S at the leaves of a balanced binary search tree, sorted by the values of W_p. At each node u of this tree, we store a secondary data structure as outlined below.

Given a query point p, we want to find all $q \in S_{p,>}$ such that $(p_1 - q_1)^2 + (p_2 - q_2)^2 \leq (\sigma W_p)^2$. 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 $(p_1 - q_1)^2 + (p_2 - q_2)^2 \leq (\sigma W_p)^2$. These are all points q that are

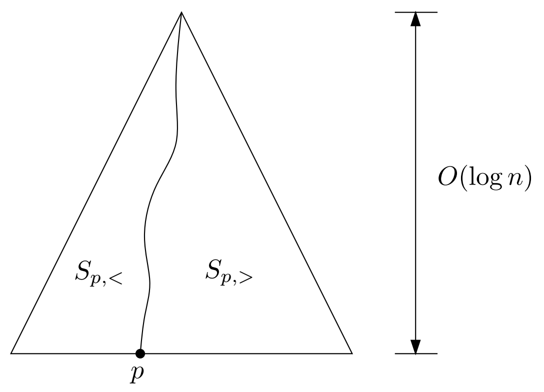

Fig. 3. Binary tree associated to the point set.

in the circle with center p and radius σW_p. So we store at u a data structure that supports these queries. Thus the time to construct the σ-local graph is $O(n \log n)$ times the time for a circular range reporting query.

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 each level encompasses a linear number of nodes, the total time complexity to construct $G_\sigma(S)$ is thus $O(n \log^3 n \log\log n + k)$. $\square$

3.3 σ-Spectrum

Lemma 6 *Given a point set S in $\mathbb{R}^2$ and a constant c, 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 [15], 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 [8] 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.

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 $n+K$ processors and only algebraic predicates.

This immediately allows us to use the parametric search technique of Megiddo [15] to find the smallest σ such that $G_\sigma(S)$ contains at least $K = n \log^3 n \log\log n$ edges in time $O(n \log^{O(1)} n)$. $\square$

Theorem 7 *Using $O(n \log^{O(1)} n)$ space and preprocessing time, we can compute $G_\sigma(S)$ for any set S of n points in $\mathbb{R}^2$ and any σ in $O(k)$ time, where k is the number of edges in the resulting graph.*

PROOF. We can preprocess efficiently σ-local graphs because the size of the graph is a positive monotone function of σ: if σ grows, edges are added and never removed.

We first note that if the output has more than $n \log^3 n \log\log n$ edges, we can construct the graph using the algorithm given in Theorem 5.

The preprocessing phase consists in constructing a graph with $n \log^3 n \log\log n$ edges, with the method given by Lemma 6. We sort the edges of the resulting graph with respect to σ and we store the edges in a list. All of this can be achieved in $O(n \log^{O(1)} n)$ time.

To answer a query, we simply output every edge in the list up to the corresponding value of σ. $\square$

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 Completeness

A first simple question when we deal with different values of σ is to check whether or not the resulting graph is complete.

Let $\mathrm{NN}(p)$ and $\mathrm{FN}(p)$ denote the nearest and the farthest Euclidean neighbor of a vertex p in the set S.

Lemma 8 *The graph $G_\sigma(S)$ is not the complete graph on S if and only if there is a point p in S such that $|p\mathrm{FN}(p)|/|p\mathrm{NN}(p)| \geq \sigma$.*

PROOF. Assume that p is a point in S such that $|p\mathrm{FN}(p)|/|p\mathrm{NN}(p)| \geq \sigma$. Let $q = \mathrm{FN}(p)$. Then $\mathrm{NN}(p)$ is contained in the ball with center p and radius $|pq|/\sigma$. Thus, (p, q) is not an edge in $G_\sigma(S)$ and $G_\sigma(S)$ is not the complete graph.

To prove the converse, assume that (p, q) is not an edge in $G_\sigma(S)$. Then

$$|p\mathrm{NN}(p)| \leq |pq|/\sigma \leq |p\mathrm{FN}(p)|/\sigma,$$

or

$$|q\mathrm{NN}(q)| \leq |pq|/\sigma \leq |q\mathrm{FN}(q)|/\sigma.$$

Hence, p or q satisfies the condition in the lemma. $\square$

This lemma implies that the largest value of σ for which $G_\sigma(S)$ is not the complete graph on S is given by

$$\sigma = \max_{p\in S} \frac{|p\mathrm{FN}(p)|}{|p\mathrm{NN}(p)|}.$$

Theorem 9 *Given a set of n points in $\mathbb{R}^2$, the maximum σ such that the corresponding σ-local graph is not complete can be found in $O(n \log n)$ time.*

PROOF. The nearest and farthest neighbor graph of the set of points can be constructed in $O(n \log n)$ time [17]. We find the maximum of the ratio $\frac{|p\mathrm{FN}(p)|}{|p\mathrm{NN}(p)|}$

for every point of S in linear time, and using Lemma 8 we know that this gives us the largest value of σ such that $G_\sigma(S)$ is not complete. □

4.2 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 10 *Given a set S of 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. Therefore, we use an algorithm similar to the one used for Lemma 6. 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 and Lemma 6.

As above, we use a binary search tree to store the vertices according to their weights. Then, we check for every vertex p if it is connected to any other vertex, either in $S_{p,>}$ or $S_{p,<}$.

For the vertices in $S_{p,>}$, we use the same method which consists in circular range queries. We do not need to effectively report all points, we just want to ensure that at least one of the vertices is connected.

For the vertices in $S_{p,<}$, the edge q leading to the lowest σ is the closest from p, as we know that σ only depends on the distance $|pq|$ and W_p. Given a Voronoi diagram of the point set $S_{p,<}$, finding the optimal q is a point location problem: by definition it is the point whose Voronoi cell contains p.

In the preprocessing phase, we construct the binary tree and 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.

After the preprocessing phase, we apply the same process for every point p in S: for every node in the path from the root to the leaf of the tree representing that point, we check if there is an edge from p to any item of $S_{<,p}$ or $S_{>,p}$. The length of the path is in $O(\log n)$ and for each node, we must either locate p in a Voronoi diagram in $O(\log n)$ time or perform the circular range query. In conclusion, we can decide if a point is connected to p in $O(\log^2 n)$ time for a given σ.

To find the lowest σ, we use the parametric search technique of Megiddo [15]. The Voronoi diagrams do not depend on σ. Thus, the complete algorithm will be as follows: construct the tree and the Voronoi diagrams in $O(n \log^2 n)$ time; compute in parallel the optimal value for σ by querying for each point if it is isolated.

The canonical sets have size $O(n \log n)$ in total, and we run our parallel algorithm on $P = n$ processors. The sequential time T_s, that is the time to check if there is an isolated vertex for a given σ, is $O(n \log^2 n)$; the parallel time T_p is $O(\log^2 n)$. The total complexity to find the minimum value of σ such that there is no isolated vertex is then $O(PT_p + T_pT_s \log P)$, which enumerate to $O(n \log^5 n)$. $\square$

4.3 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 [5, 16] for finding the minimum spanning tree.

Theorem 11 *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 consists of maintaining a list of all the components already identified and to update σ such as to connect progressively every component to others. We begin by computing the nearest neighbor of every vertex in a specific metric: the distance between any pair of vertices is the minimum value of σ such that the pair is connected. This gives, for each vertex, the first vertex to which it connects as σ grows; thus we have an initial value for σ and a list of at most $n/2$ different components.

Then we proceed as follows: 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 4.2, with Voronoi diagrams and support for circular range queries.

In the binary tree, we can easily find sets covering every vertices 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 represents covering sets for all the other components.

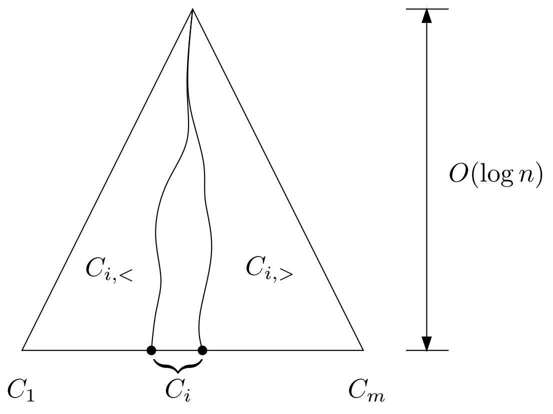

Fig. 4. Binary tree associated to the components.

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

This reduces the number of components, and 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 $O(n \log^7 n)$. $\square$

5 Planar Embedding

The last property we analyze is whether the σ-local graph obtained is a planar embedding. The following algorithm allows us to find the largest value of σ for which this is the case, and also constructs the corresponding graph in the mean time.

Theorem 12 *Given a set S of n points in $\mathbb{R}^2$, the planar σ-local graph with maximum σ can be constructed in $O(n \log^{O(1)} n)$ time.*

PROOF. Using Lemma 6, 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 weight in $O(n \log^{O(1)} n)$ time 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 [4] 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. □

6 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.

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

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

[6] 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.

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

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

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

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

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

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

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

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

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

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

[17] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics. John Wiley & Sons Ltd., Chichester, UK, 2nd edi-

tion, 2000.

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

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

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