Optimal and Suboptimal Robust Algorithms for Proximity Graphs*

Ferran Hurtado[†] *Giuseppe Liotta*[‡] *Henk Meijer*[§]

Abstract

Given a set of n points in the plane, any β-skeleton and $[\gamma_0, \gamma_1]$-graph can be computed in quadratic time. The presented algorithms are optimal for β values that are less than 1 and $[\gamma_0, \gamma_1]$ values that result in non-planar graphs. We show a numerically robust algorithm that computes Gabriel graphs in quadratic time and degree 2. We finally show how a β-spectrum can be computed in optimal $O(n^2)$ time.

1 Introduction and Overview

A typical approach to extracting a shape from a given set P of n points is to compute a *proximity graph* of P, i.e. a geometric graph whose vertices are elements of P and where the edges are straight-line segments connecting pairs of points. In a proximity graph of P two points are connected by an edge if and only if their *region of influence* is *empty*, i.e. it does not contain any other element of P. The region of influence of two points u and v of P is a region of the plane that describes a neighbourhood of u and v; the emptiness of the region of influence witnesses that u and v are close enough to each other to be connected by an edge. Depending on the application context, different definitions of region of influence and of corresponding proximity graphs have been proposed in the literature. A comprehensive survey is given by Jaromczyk and Toussaint [10]; here we only recall some of the most widely studied proximity graphs.

Two continuous hierarchies of proximity graphs that includes Gabriel graphs and relative neighbourhood graphs as special cases were first defined in the computational morphology context by Kirkpatrick and Radke [13]. The elements of these infinite families of proximity graphs are called *circle- and lune-based β-skeletons* and are defined by considering a continuous family of regions of influence indexed by a single real positive parameter β. Another parameterized family of proximity graphs, known as *γ-graphs* which unifies circle-based β-skeletons, convex hulls, and Delaunay triangulations was defined by Veltkamp [23]. While β-skeletons and γ-graphs are used to describe the internal shape of a set of points, *α-hulls*, defined by Edelsbrunner, Kirkpatrick, and Seidel [7], are a family of proximity graphs that can be used to describe the spectrum of external shapes of a set of points.

*Research supported in part by the CNR Project "Geometria Computazionale Robusta con Applicazioni alla Grafica ed al CAD", the project "Algorithms for Large Data Sets: Science and Engineering" of the Italian Ministry of University and Scientific and Technological Research (MURST 40%), by Gen. Cat. SGR000356 and MEC-DGES-SEUID PB98-0933, Spain, and by NSERC, Canada.

[†]Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain. Email: `hurtado@ma2.upc.es`.

[‡]Dipartimento di Ingegneria Elettronica e dell'Informazione, Università degli Studi di Perugia, Via Duranti, 06125 Perugia, Italy. Email: `liotta@diei.unipg.it`.

[§]Dept. of Computing and Information Science Queen's Univ., Kingston, Ont., Canada. Email: `henk@cs.queensu.ca`.

This paper is devoted to the study of efficient algorithms for computing β-skeletons and γ-graphs. In order to better explain our contribution, we briefly review some literature about proximity graphs and then list our results. Existing algorithms that compute proximity graphs can be classified according to whether they assume the *Fixed Proximity Scenario* or whether they assume the *Variable Proximity Scenario.*

Fixed Proximity Scenario: In the fixed proximity scenario the input of the problem is a set of points and a definition of closeness between pairs of points; the output is a geometric graph such that two vertices are adjacent if and only if they satisfy the given definition of proximity.

Variable Proximity Scenario: In this scenario it is not known a priori what closeness measure to use: the application context requires to consider several different definitions of closeness in order to choose the best suited one [19]. In this scenario, the input of the problem is a set of points and a set of definitions of closeness between pairs of points; the output is a set of geometric graphs describing the different definitions of proximity.

In the Fixed Proximity Scenario, optimal $O(n \log n)$ time algorithms are known for computing lune-based β skeletons when $\beta = 1$ and when $\beta = 2$ [17, 22]. For $1 < \beta < 2$, optimal $O(n \log n)$ time algorithms are described in [11, 14]. To our knowledge, there exist only suboptimal algorithms that compute the lune-based β-skeleton when $0 \leq \beta < 1$: the fastest algorithm that we know for this problem requires $O(n^{2.5} \log n)$ time [20]. For values of β in the range $(2, \infty)$, a (suboptimal) $O(n^2)$ time algorithm for lune-based β-skeletons is described in [13, 19]. As for circle-based β-skeletons, an optimal $O(n \log n)$ time algorithm is given in [13] for all values of β in the interval $[1, \infty]$, while the same $O(n^{2.5} \log n)$ time algorithm of [20] applies for values of β in the interval $[0, 1)$. The problem of computing γ-graphs has been studied in [23], where an $O(n^3)$ time algorithm is described for the general case. Under the assumption that no four points are cocircular and that the chosen value of γ gives rise to a γ-graph with no edge crossings, an optimal $O(n \log n)$ time algorithm is also presented in [23].

In the Variable Proximity Scenario, a key problem is that of encoding the entire spectrum of the empty neighbourhoods that can be found in the point set. Given this information, it is easy to compute different proximity graphs with an output sensitive strategy. A first $O(n^3)$ time algorithm for computing the β-spectrum of a set of n points is given in [13]; the algorithm can be used both for the case that the β-neighbourhood is lune-based and the case that it is circle-based. Very recently, a new algorithm for the computation of the lune-based β-spectrum was presented in [21]. The algorithm in [21] requires $O(n^2 \log n + \rho)$ time, where ρ is a parameter that depends upon the geometry of the point set and it can be $O(n^3)$ for some problem instances (ρ is the size of the so called *witness set*; see [21] for more details).

In this paper we present two simple algorithmic strategies that can be applied to solve a number of proximity problems both in the Fixed Proximity Scenario and in the Variable Proximity Scenario. Our results can be listed as follows.

- We exhibit an $O(n^2)$-time algorithm for the computation of the (lune-based or circle-based) β-skeleton of a set of n points in the plane. This is worst case optimal when $0 \leq \beta < 1$ since for these values of β the β-skeleton can have $\Theta(n^2)$ edges. As described above, the previously known time bound for this problem is $O(n^{2.5} \log n)$ [20].

- We extend our technique to γ-graphs and present an optimal $O(n^2)$ time algorithm for their computation in the Fixed Proximity Scenario. The previously known algorithm requires $O(n^3)$ time [23].

- We give an $O(n^2)$-time algorithm for the computation of the circle-based β-spectrum. Our algorithm is optimal, (the size of the β-spectrum is n^2) and improves over the previously known $O(n^3)$ time algorithm [13].
- We extend the technique of the previous item and obtain an optimal $O(n^2)$-time algorithm for the computation of the lune-based β-spectrum. This result improves over the $O(n^2 \log n + \rho)$-time algorithm in [21].
- As a further application of our technique we show the first algorithm that computes the Gabriel Graph of a set of n points in $O(n^2)$ time and requires only double precision integer arithmetic computations. We remark that the optimal time $O(n \log n)$-time algorithm described in the literature relies on the computation of the Delaunay triangulation which may require an arithmetic precision four times the one used to represent the input data [5]. References where the performance of geometric computations is measured also in terms of the required arithmetic precision include [2, 3, 4, 16].

The remainder of the paper is organized as follows. Preliminaries are in Section 2. A unifying approach to the computation of β-skeletons and γ-graphs is presented in Section 3. In the same section the robust algorithm for Gabriel graphs is described. Section 4 deals with the computation of the β-spectrum. Finally, conclusions and open problems can be found in Section 5.

2 Preliminaries

In this section we introduce some notation and definitions that we use in subsequent sections. Let P be a set of points in the plane and let u and v be two points of P. The Euclidean distance between u and v is denoted by $d(u,v)$. The *arc* (u,v) is the directed line segment from u to v, and the *edge* (u,v) is the undirected line segment from u to v. Consider an arbitrary circle of radius r through u and v and let C be one of the two circular arcs connecting u to v on this circle.

We associate with C a value that we will call the *relative curvature* of C, which is in fact related to the curvature of C or to its reciprocal thereof. If the length of C is larger than πr, we define the relative curvature β of C as $2r/d(u,v)$. If the length of C is smaller than or equal to πr, we define the relative curvature β of C as $d(u,v)/r$. If $r = \infty$, then C is either the edge (u,v) or consists of two infinite half lines with endpoints u and v. We say that the relative curvature of the edge (u,v) is 0, while the relative curvature of the two infinite half lines is equal to ∞.

Let C be a circular arc of relative curvature β that lies to the right of an arc (u,v). We call the area bounded by edge (u,v) and C the *right β-region* of the arc (u,v) and denote it as $C_r(u,v,\beta)$. The region $C_r(u,v,\beta)$ is assumed to include the edge (u,v) but not C. For $\beta = \infty$, we define $C_r(u,v,\infty)$ as the open half plane to the right of the arc (u,v), plus the edge (u,v). Finally $C_r(u,v,0)$ is the empty set. See Figure 1 for an illustration. Given two points u and v in a point set P, we say that the right β-region of the arc (u,v) is *empty* if it does not contain any point of P other than u and v. For example, in Figure 1 both $C_r(u,v,1)$ and $C_r(v,u,\frac{1}{2})$ are empty.

We associate with the arc (u,v) a value that we call the *right β-value* of the arc (u,v) and denote it as $\beta_r(u,v)$. The right β-value of (u,v) is the largest value of β such that $C_r(u,v,\beta)$ is empty. The *β-value* of an edge (u,v) is the minimum of $\beta_r(u,v)$ and $\beta_r(v,u)$ and is denoted as $\beta(u,v)$. For example, the right β-value of (u,v) in Figure 1 is $\beta_r(u,v) = 1$, the right β-value $\beta_r(v,u) = \frac{1}{2}$, and the β-value is $\beta(u,v) = \frac{1}{2}$.

The *β-spectrum* of a set P of points [13] is the set of all edges spanned by the set P where each edge (u,v) is labeled with its β-value. The *circle-based β-neighbourhood* [13] of two points u and v

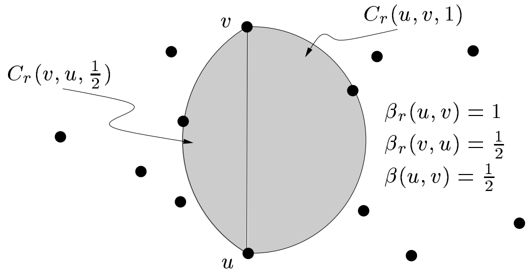

Figure 1: The right β-values of (u, v) and (v, u) and the β-value of (u, v).

is equal to $C_r(u, v, \beta) \cup C_r(v, u, \beta)$. The *circle-based β skeleton* of P is a proximity graph such that an edge (u, v) belongs to the graph if and only if $C_r(u, v, \beta) \cup C_r(v, u, \beta)$ is empty. For example in Figure 1, (u, v) is an edge of the β-skeleton for $\beta = \frac{1}{2}$; (u, v) is not in the β-skeleton for $\beta > \frac{1}{2}$.

For $0 \leq \beta \leq 1$, the *lune-based β neighbourhood* is the same as the circle-based β neighbourhood. For $\beta > 1$, the lune-based β-neighbourhood of points u and v is the intersection of two disks of radius $\beta d(u, v)/2$ whose centres lie on the line through (u, v); one of the disks has u in its interior and v on its boundary, the other one has v in its interior and u on its boundary. For an illustration, see Figure 2.

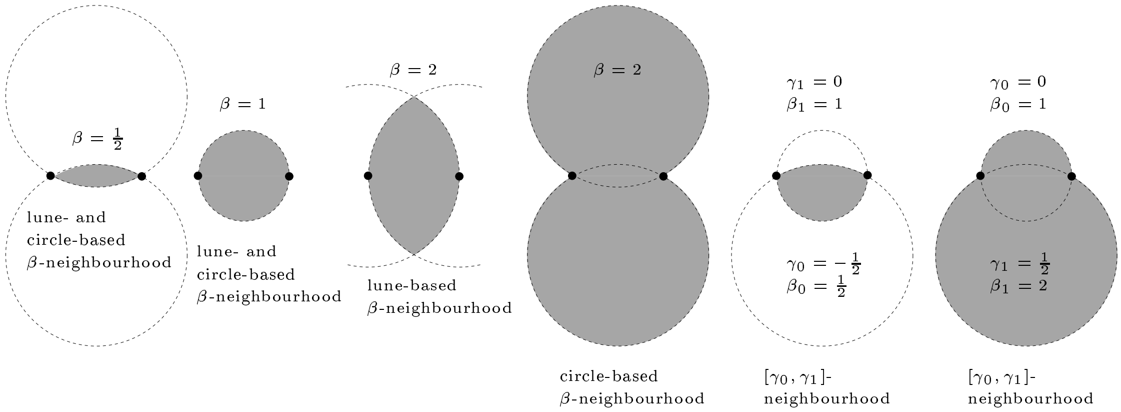

Figure 2: β- and $[\gamma_0, \gamma_1]$-neighbourhoods.

In [23] Veltkamp introduces *γ-graphs* as a parameterized family of proximity graphs which include circle-based β-skeletons as special cases and are defined in terms of two *γ-parameters*, named γ_0 and γ_1. The γ-graph of P is also denoted as the $[\gamma_0, \gamma_1]$-graph of P. Let γ_0 and γ_1 be two real values such that $|\gamma_0| \leq |\gamma_1| \leq 1$ and let u, v be a pair of points of P. The values γ_0 and γ_1 define two *$[\gamma_0, \gamma_1]$-neighbourhoods* of (u, v). The $[\gamma_0, \gamma_1]$-graph of P is a proximity graph such that any two vertices u, v are adjacent if and only if at least one of the two $[\gamma_0, \gamma_1]$-neighbourhoods of (u, v) is empty. The two $[\gamma_0, \gamma_1]$-neighbourhoods of (u, v) can be defined in terms of right β-regions: the $[\gamma_0, \gamma_1]$-neighbourhoods of (u, v) are $C_r(u, v, \beta_0) \cup C_r(v, u, \beta_1)$ and $C_r(v, u, \beta_0) \cup C_r(u, v, \beta_1)$, where the correspondence between the pair (β_0, β_1) and (γ_0, γ_1) is as follows. If $\gamma_i \geq 0$ then $\beta_i = 1/(1 - \gamma_i)$ and if $\gamma_i \leq 0$ then $\beta_i = 1 + \gamma_i$ for $i = 0, 1$.

3 Algorithms for the Fixed Proximity Scenario

We first present an algorithm for the following problem. Let P be a set of points in the plane and let β be a fixed value such that $0 < \beta < \infty$. For each pair of points u, v of P we ask ourselves whether or not the right β-region of the arc (u, v) contains an element of P. We call this problem the *right β-region problem*. We shall exploit our solution to the right β-region problem to efficiently compute β-skeletons, γ-graphs, and Gabriel graphs with low arithmetic degree.

3.1 The Right β-region Problem

Our algorithm for the right β-region problem assigns each arc (u, v) a label according to whether the right β-region of (u, v) is empty or not. The arc (u, v) is labeled *Yes* if $C_r(u, v, \beta)$ contains no points of P and it is labeled *No* otherwise. A technique similar to ours has been used by Beirouti and Snoeyink [1] for computing triangle emptiness in the context of LMT heuristics for minimum weight triangulations.

`Algorithm Right` β`-region`

`Step 1:` For each point $p \in P$ compute an ordered list of the remaining points, sorted in radial order around p in clockwise direction. Do `steps` 2, 3 and 4 for each point $p \in P$.

`Step 2:` Let $p_0, p_1, \ldots, p_{n-2}$ be the ordered sequence of points of $P - \{p\}$ radially sorted around p in clockwise direction. Each point p_i will get a label *No* or *Maybe*. For each p_i we will store an index called $PrevMaybe$, which will be index of a previous point labeled *Maybe*. Let $lmi = -1$, where lmi is the index of the last point labeled *Maybe* and -1 indicates that lmi is undefined. Arithmetic in the indices is done $\mathrm{mod}(n-1)$.

For $i = 0, 1, 2, \ldots, n-2$ do:

`Step 2.1:` (For an illustration, see Figure 3.)
If $p_{i+1} \notin C_r(p, p_i, \beta)$ then

`Step 2.1.1.1:` Label p_i with a label *Maybe*.

`Step 2.1.1.2:` $PrevMaybe$ of p_i becomes lmi.

`Step 2.1.1.3:` $lmi = i$.

else

`Step 2.1.2.1:` Label p_i with a label *No*.

`Step 2.1.2.2:` Let $j = lmi$. If $j \geq 0$ and $p_{i+1} \in C_r(p, p_j, \beta)$ then label p_j with label *No*, set lmi equal to the $PrevMaybe$ value of p_j and repeat `Step 2.1.2.2`.

`Step 3:` Let t the the largest index such that the clockwise angle between arcs (p, p_0) and (p, p_t) is $< \pi$. Scan $p_0, p_1, \ldots, p_t$; if point p_i $(0 \leq i \leq t)$ is labeled *Maybe* assign the label *Yes* to the arc (p, p_i), else assign the label *No* to this arc.

`Step 4:` : Let t be as defined in `Step 3`. Renumber the points $p_0, p_1, \ldots, p_{n-2}$ such that p_0 becomes p_{t+1}, p_1 becomes p_{t+2}, ..., p_{n-2} becomes p_{n+t-1}. Then repeat `Steps 2` and `3`.

`End of Algorithm Right` β`-region`

The following lemmas prove the correctness of the above algorithm and show that its time complexity is $O(n^2)$. The proof of the first lemma follows from elementary geometry.

Lemma 1 *If $i < j$ and the clockwise angle between arcs (p, p_i) and (p, p_j) is less than π and if $p_j \notin C_r(p, p_i, \beta)$ then $C_r(p, p_i, \beta) \cap C_r(p, p_j, \infty) \subset C_r(p, p_j, \beta)$. If $i < j$ and the clockwise angle between arcs (p, p_i) and (p, p_j) is $\geq \pi$, then $p_j \notin C_r(p, p_i, \beta)$.*

Lemma 2 *After* `Algorithm Right` β`-region` *has performed* `Step 2.1` *for a particular value of i, we have that for each value of h with $0 \leq h \leq i$ for which p_h is labeled* Maybe*, $C_r(p, p_h, \beta)$ does not contain any points of $\{p_{h+1}, p_{h+2}, \ldots, p_{i+1}\}$.*

Proof: The lemma is true for $i = 0$. Assume that the lemma holds for $i = 0, 1, \ldots, m-1$. Let $i = m$ and let $0 \leq h \leq m$. If p_m is labeled *Maybe* then the statement is true for $h = m$ because $p_{i+1} \notin C_r(p, p_m, \beta)$.

Consider a point p_h labeled *Maybe* such that $0 \leq h \leq m-1$. By the inductive assumption, $C_r(p, p_h, \beta)$ does not contain any points from $\{p_{h+1}, p_{h+2}, \ldots, p_i\}$. We will show that either the label of p_h changes to *No* or $p_{i+1} \notin C_r(p, p_h, \beta)$.

If p_i gets a label *Maybe* then $p_{i+1} \notin C_r(p, p_i, \beta)$ and by Lemma 1, $p_{i+1} \notin C_r(p, p_h, \beta)$. If p_i gets a label *No* then $p_{i+1} \in C_r(p, p_i, \beta)$. The algorithm repeatedly finds the most recent point p_j that has been assigned a *Maybe* label, and changes this label to *No* if $p_{i+1} \in C_r(p, p_j, \beta)$. If `Step 2.1.2.2` stops with $h \leq j$, we have that $p_{i+1} \notin C_r(p, p_j, \beta)$, so by Lemma 1, $p_{i+1} \notin C_r(p, p_h, \beta)$. If `Step 2.1.2.2` stops with $j < h$, then $p_{i+1} \in C_r(p, p_h, \beta)$ and the algorithm will change the label of p_h to *No*. Therefore this lemma holds. $\square$

Lemma 3 *The* `Algorithm Right` β`-region` *assigns* Yes *labels to those arcs (p, p_i) for which $C_r(p, p_i, \beta)$ is empty.*

Proof: If $C_r(p, p_i, \beta)$ contains a point p_j, then the clockwise angle between p_i and p_j is $< \pi$. So Lemma 2 shows that in `Step 3` the algorithm correctly assign *Yes* or *No* labels to all arcs (p, p_i), where p_i lies in the half plane $C_r(p, p_0, \infty)$. By the same argument, the remaining arcs are assigned *Yes* or *No* labels in `Step 4`. $\square$

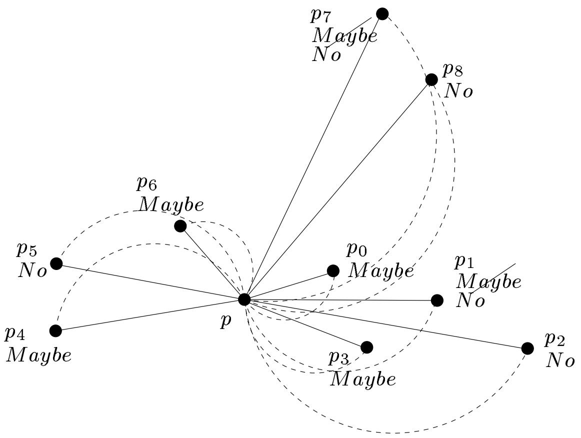

Figure 3: The *Maybe*/*No* labeling strategy of `Step 2` of `Algorithm Right` β`-region`.

Theorem 1 *Let P be a set of n points in the plane.* `Algorithm Right` β`-region` *solves the Right β-region Problem for P in $O(n^2)$ time.*

Proof: Correctness follows from Lemma 3. The radial sorting of `Step 1` can be done in $O(n^2)$ time by the algorithm of [18] or the alternative algorithm in [6]. Since in `Step 2` each point receives a label *Maybe* at most once, this step requires linear time. Similarly `Steps 3` and `4` require linear time. Since `Steps 2`, `3` and `4` are executed n times, `Algorithm Right` β`-region` has an $O(n^2)$ time complexity. □

3.2 Computing Proximity Graphs

We will now use `Algorithm Right` β`-region` to compute proximity graphs. We first consider β-skeletons for values of β such that $0 \leq \beta < 1$. In this interval the β-neighbourhood is the same for lune-based and circle-based graphs; therefore in the proof of the next theorem we do not distinguish between lune-based and circle-based β-skeletons.

Theorem 2 *Let P be a set of n points in the plane and let $0 \leq \beta < 1$. There exists an algorithm that computes the circle-based and the lune-based β-skeleton of P in optimal $O(n^2)$ time.*

Proof: If $\beta = 0$, the β-skeleton of P is the complete graph and can be easily computed in $O(n^2)$ time by connecting all pairs of points. If $\beta > 0$, we compute the β-skeleton of P as follows. We first execute `Algorithm Right` β`-region` on P. Then we compute all edges (u, v) of the β-skeleton by checking if both the arc (u, v) and the arc (v, u) have been labeled *Yes* by `Algorithm Right` β`-region`. The correctness of the algorithm and the bound on its time complexity are a consequence of Theorem 1. □

Theorem 3 *Let P be a set of n points in the plane and let γ_0, γ_1 be a pair of real values such that $|\gamma_0| \leq |\gamma_1| \leq 1$. There exists an algorithm that computes the $[\gamma_0, \gamma_1]$-graph of P in optimal $O(n^2)$ time.*

Proof: We compute the $[\gamma_0, \gamma_1]$-graph of P by the following procedure. First the value of β_0 and the value of β_1 corresponding to γ_0 and γ_1 are computed (See Section 2). We execute `Algorithm Right` β`-region` on P twice: once for $\beta = \beta_0$ and a second time for $\beta = \beta_1$. A pair u, v of points is connected by an edge in the $[\gamma_0, \gamma_1]$-graph if one of the following two events happens:

1. The arc (u, v) has been labeled *Yes* by `Algorithm Right` β`-region` when $\beta = \beta_0$ and (v, u) has been labeled *Yes* when $\beta = \beta_1$; or

2. The arc (u, v) has been labeled *Yes* by `Algorithm Right` β`-region` when $\beta = \beta_1$ and (v, u) has been labeled *Yes* when $\beta = \beta_0$.

The bound on the time complexity is a consequence of Theorem 1. □

As a further application of our approach to computing proximity graphs in the fixed proximity scenario, we consider circle-based β-skeletons for values of β such that $\beta \geq 1$ and for β-regions that are closed sets. We define the *closed right β-region* $C_r(u, v, \beta)$ as follows. Let C be a circular arc of relative curvature β that lies to the right of an arc (u, v). The closed area bounded by edge (u, v) and C is the closed right β-region of the arc (u, v).

It is straightforward to extend `Algorithm Right` β`-region` to deal with closed right β regions: the algorithm does not need to be modified except that the test about whether a point is inside the closed right β region would give a positive answer when the point is on C.

Theorem 4 *Let P be a set of n points in the plane and let $0 \leq \beta \leq \infty$. There exists an algorithm that computes the open and closed circle-based β-skeleton of P in $O(n^2)$ time.*

Proof: The proof for the cases $\beta = 0$ and $\beta = \infty$ is straightforward. For $0 < \beta < \infty$ the algorithm to compute a β-skeleton is an extension of `Algorithm Right` β`-region`. □

When $\beta = 1$ and the β neighbourhood is a closed set, the β-skeleton of P is the Gabriel graph of P [13]. It follows from Theorem 4 that the Gabriel graph of a set of n points can be computed in $O(n^2)$ time. This result is not very surprising on its own since an optimal $O(n \log n)$ algorithm for Gabriel graphs (and also for every circle-based β-skeleton with $\beta \geq 1$) is known [8, 13]. However, this changes if we revisit existing algorithms in terms of their robustness.

The optimal algorithm for Gabriel graphs is based on first computing the Delaunay triangulation and then on deleting the edges that are not Gabriel edges. Unfortunately, the implementation of asymptotically optimal algorithms that compute Delaunay triangulations and correctly handle input degeneracies is not an easy task in practice (see, e.g. [9, 12]). A numerically robust code for Delaunay triangulations must evaluate the sign of irreducible polynomials of algebraic degree four, therefore requiring a numerical precision at least four times the one needed for representing the input data; evaluating the sign of these polynomials is needed to execute geometric tests which determine whether a point is contained in the disk defined by other three points.

On the other hand, it is a trivial task to design an $O(n^3)$ time algorithm that computes the Gabriel graph of a set of points based on only distance comparisons. In this case, the algebraic degree of the polynomials to evaluate is only two, and a robust implementation can be accomplished by using double precision integer arithmetic. Thus, we can identify a trade-off between time complexity and numerical precision required by algorithms that compute Gabriel graphs. In the next theorem we show how to reduce the trade-off.

We adopt the *degree model* of computation introduced in [2, 16] and analyze the performance of `Algorithm Right` β`-region` in terms of required numerical precision to compute Gabriel graphs. In this model of computation the robustness of a geometric algorithm is evaluated by looking at the irreducible polynomial of highest algebraic degree whose sign is evaluated by the algorithm during its execution. This quantity is called the *degree* of the algorithm and is a measure of the numerical precision that a robust implementation would require to guarantee correct outputs independent of degenerate configurations of the input data. Namely, if an algorithm has degree d and its input variables are b-bit integers, then all test computations can be carried out with at most $d(b + O(1))$ bits. A problem has degree d if any algorithm that solves it has degree at least d.

By analyzing the algebraic degree of the geometric tests in the elegant and simple algorithm by Overmars and Welzl [18] we can conclude the following.

Lemma 4 *Let P be a set of n points in the plane. The algorithm by Overmars and Welzl [18] solves the problem of reporting for each $u \in P$ all points of $P - \{u\}$ radially sorted around u in $O(n^2)$ time and degree* 2.

Proof: The only geometric tests performed by the radial sorting algorithm of Overmars and Welzl are `which-side-tests` that verify for any three given input points (p, r, q) whether q is on the right hand-side or on the left hand-side of the line through p and q. This test corresponds to the evaluation of an irreducible polynomial of algebraic degree 2. □

The next theorem shows the first algorithm that computes the Gabriel graph of a set of n points in optimal degree and $o(n^3)$ time.

Theorem 5 *Let P be a set of n points in the plane. There exists an optimal degree* 2 *algorithm that computes the Gabriel graph of P in $O(n^2)$ time.*

Proof: We compute the Gabriel graph of P by executing `Algorithm right` β`-region` on P, with $\beta = 1$ and closed right β-regions. For a pair u, v of points in P, if both the closed right 1-region of arc (u, v) and the closed right 1-region of arc (v, u) are empty, then we connect u and v in the Gabriel graph. The correctness of the algorithm and its time complexity are a consequence of Theorem 4.

`Algorithm Right` β`-region` executes geometric tests in `Step 2.1` where it checks whether p_{i+1} is in the closed disk having p, p_i as antipodal points, and in `Step 3` where it computes the largest index t such that the angle between arcs (p, p_0) and (p, p_t) is less than π.

The geometric test of `Step 2.1` is an `in-circle test` which given any three points p, q, r it checks whether r is on, inside, or outside the disk whose diameter is the segment pq. The `in-circle test` can be done by computing the centre c of the disk having p, q as antipodal points, and by comparing $d(p, c)$ with $d(r, c)$. This corresponds to determining the sign of an irreducible polynomial of algebraic degree 2 whose variables require at most $b + 1$ bits to be represented. Also, `Step 3` computes the index p_t by executing (a linear number of) `which-side tests` that can be executed with degree 2. Therefore the degree of the algorithm is 2. Finally, the algorithm is degree optimal because, as shown in [15], the problem of computing the Gabriel graph of a set of points has degree 2. $\square$

4 Algorithms for the Variable Proximity Scenario

In this section we consider the problem of computing the β-spectrum of a set P of points in the plane. Similarly to what we did in the previous section, we address the problem by first solving a different problem. For each arc (u, v) we compute the right β-value $\beta_r(u, v)$ and then compute the β-value of u, v as the minimum of $\beta_r(u, v)$ and $\beta_r(v, u)$.

The section is organized as follows. We first introduce some notation, define a set of points with special properties, and study some basic properties of this set of points. We are then in a position to present the algorithm and its proof of correctness.

Let u, v and w be three points in the plane and let $C(u, v, w)$ be the disk through u, v and w. We denote with $C'(u, v, w)$ the subset of $C(u, v, w)$ bounded by chord (u, w) and the circular arc from u, through v ending at w. $C'(u, v, w)$ includes the chord (u, w), but not the circular arc from u to w.

From here until the beginning of Lemma 9, we assume that $P = \{p, p_0, p_1, \ldots, p_{n-1}\}$ has the following property. Let C be a circle through p. Let l_0 be a line through p and tangent to C. Assume that l_0 is horizontal and that C lies above l_0. The points $\{p_0, p_1, p_2, \ldots p_{n-1}\}$ lie above l_0 and are counterclockwise radially sorted around p. The point p_0 lies on C and the points $\{p_1, p_2, \ldots p_{n-1}\}$ lie on or outside C. Let B be a subset of P. We say that $B = \{p, p_0, p_1, p_2, \ldots p_k\}$ is an *increasing set* if p_j is on or outside $C(p, p_{i-1}, p_i)$ for all $0 \leq j \leq k$ and $0 < i \leq k$. An increasing set of points is depicted in Figure 4.

Let l be an open halfline starting at p, such that l lies above l_0 and such that the points $\{p_0, p_1, p_2, \ldots p_k\}$ lie to the right of l. Let a_0 be the intersection point of C and l and let a_i be the intersection point of l and circle $C(p, p_{i-1}, p_i)$. We say that $a_i < a_j$ if $d(p, a_i) < d(p, a_j)$. We make the following observation:

Property 1 *A set of points $B = \{p, p_0, p_1, p_2, \ldots p_k\}$ is an increasing set if and only if $a_0 \leq a_1 \leq a_2 \leq \ldots \leq a_k$.*

The following lemmas study properties of increasing sets of points. In Lemmas 5, 6 and 7 we

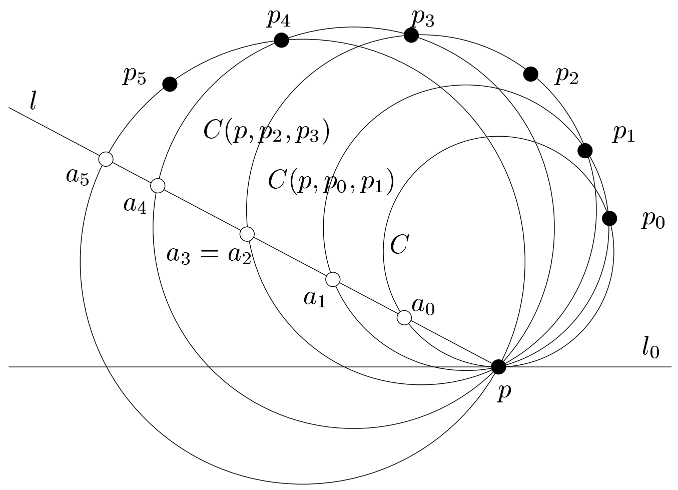

Figure 4: Increasing set $\{p, p_0, p_1, \ldots, p_5\}$.

use expressions like $C_r(p, p_k, \beta_r(p, p_k)) = C'(p, p_i, p_k)$. This means that the right β value of the arc (p, p_k) is defined by the point p_i.

Lemma 5 *Let $P = \{p, p_0, p_1, \ldots, p_k\}$ be a set of points such that $\{p, p_0, p_1, \ldots, p_{k-1}\}$ is an increasing set. Let i be the largest index such that $C_r(p, p_k, \beta_r(p, p_k)) = C'(p, p_i, p_k)$. Then $p_{j-1} \in C'(p, p_j, p_k)$ for $i < j < k$.*

Proof: If $i = k-1$ then the lemma holds trivially. So assume that $i < k-1$. Let l be an open halfline starting at p and passing through p_k. Let a_0 be the intersection point of C and l and let a_i be the intersection point and of l and circle $C(p, p_{i-1}, p_i)$. Observe first that $a_i \le p_k < a_{i+1}$. We claim that $p_{j-1} \in C'(p, p_j, p_k)$ for $i < j < k$, as illustrated in Figure 5(a) with $i = 3$, $j = 5$ and $k = 6$. Suppose there is a j with $i < j < k$ for which $p_{j-1} \notin C'(p, p_j, p_k)$. Circle $C(p, p_j, p_k)$ intersects $C(p, p_{j-1}, p_j)$ in p and p_j. Since p_{j-1} lies on or outside the circle $C(p, p_j, p_k)$ it follows that a_j lies on or inside $C(p, p_j, p_k)$, i.e. $a_j \le p_k$. So we have $a_{i+1} \le a_j \le p_k < a_{i+1}$, which is a contradiction. □

Lemma 6 *Let $P = \{p, p_0, p_1, \ldots, p_k\}$ be a set of points such that $\{p, p_0, p_1, \ldots, p_{k-1}\}$ is an increasing set. Let i be the largest index such that $C_r(p, p_k, \beta_r(p, p_k)) = C'(p, p_i, p_k)$. Then $\{p, p_0, p_1, \ldots, p_i, p_k\}$ is an increasing set of points.*

Proof: Let l be an open half-line starting at p and passing through p_k as shown in Figure 5(a). Let a_0 be the intersection point of C and l and let a_i be the intersection point and of l and circle $C(p, p_{i-1}, p_i)$. Since $C_r(p, p_k, \beta_r(p, p_k)) = C'(p, p_i, p_k)$, we have that $a_i \le p_k$. So from Property 1 it follows that $\{p, p_0, p_1, \ldots, p_i, p_k\}$ is an increasing set of points. □

Lemma 7 *Let $P = \{p, p_0, p_1, \ldots, p_k, p_{k+1}\}$ be a set of points such that $\{p, p_0, p_1, \ldots, p_{k-1}\}$ is an increasing set. If $C_r(p, p_k, \beta_r(p, p_k)) = C'(p, p_i, p_k)$ then $\{p_i, p_k\} \cap C'(p, p_j, p_{k+1}) \neq \emptyset$ for $i < j < k$.*

Proof: If p_i lies outside $C(p, p_j, p_{k+1})$, then $C(p, p_j, p_{k+1})$ intersects $C(p, p_i, p_k)$ in p and a point between p_i and p_k, so p_k lies inside $C(p, p_j, p_{k+1})$. That proves the lemma. For an illustration with $i = 3$, $j = 4$ and $k = 6$ see Figure 5(b). □

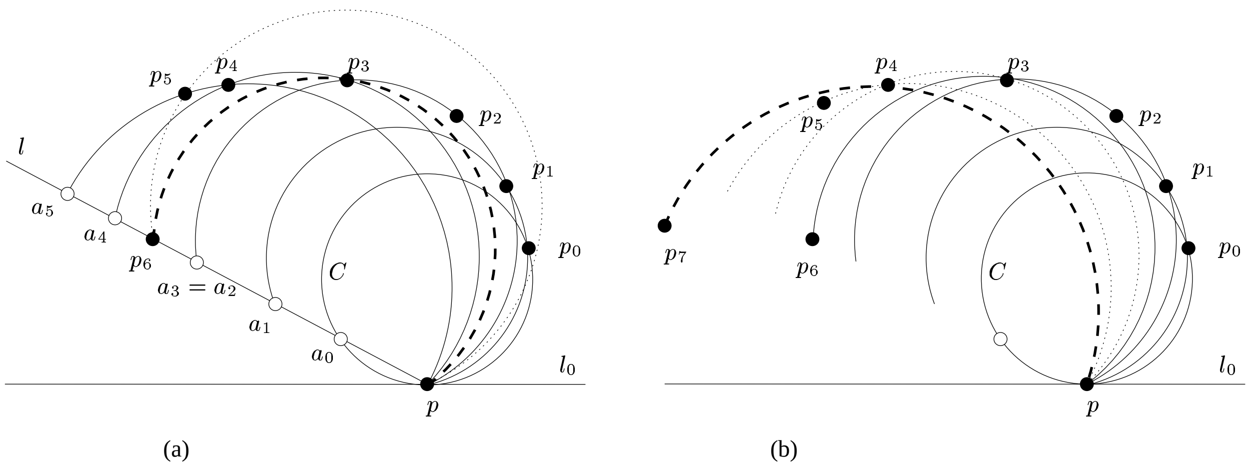

Figure 5: Illustrations for Lemmas 5, 6 and 7 with $k = 6$.

We are now ready to show how to compute the right β-values on a special set of points. Lemmas 5, 6 and 7 will be used to prove the correctness of this algorithm. We will show later how this solution can be generalized to deal with any set of points.

Let $P = \{p, p_0, p_1, \ldots, p_{n-1}\}$ be a set of points in the plane, let C be a circle through p and p_0 and let l be the line through p tangent to C. Assume that all points in $\{p_0, p_1, \ldots, p_{n-1}\}$ lie outside or on C, and on the same side of l as C does. Also assume that points $p_0, p_1, \ldots, p_{n-1}$ are radially sorted around p in counterclockwise direction. The *right β-value problem for p* is that of computing the set of right β-values $\beta_r(p, p_i)$ for $0 < i < n$.

A high level description of the algorithm is as follows. We scan all points around p in counterclockwise direction starting from p_0. Point p_0 is stored in an set of active vertices denoted as B and circularly ordered around p in a counterclockwise fashion. The right β-value of p_1 is defined by active vertex p_0, i.e. $C_r(p, p_1, \beta) = C'(p, p_0, p_1)$ and point p_1 is inserted in the set B of active vertices. At iteration k, set B contains p and those vertices that can define the right β-value of p, p_k. As we will show, if a point p_j is such that $C'(p, p_j, p_k)$ contains some other points of P, then it always contains some points of B. Also, the point p_i such that $C_r(p, p_k, \beta_r(p, p_k)) = C'(p, p_i, p_k)$ is a point of B. At the end of iteration k, p_k is added to B. The set B is always an increasing set, a fact that will be used in the proof of correctness of our algorithm.

We now give a description of the algorithm. In what follows we use the notation $pred_B(p_i)$ to denote the point in B preceding p_i.

`Algorithm Right` β`-values`

`Step 1:` Set $B = \{p, p_0, p_1\}$; let β be such that $C_r(p, p_1, \beta) = C'(p, p_0, p_1)$; store β as $\beta_r(p, p_1)$.

`Step 2:` For $k = 2, 3, \ldots, (n-1)$ do the following:

`Step 2.1:` Let q be the last point in B.

`Step 2.2:` While $(q \neq p_0)$ and $(pred_B(q) \in C'(p, q, p_k))$ do

$q = pred_B(q)$

`Step 2.3:` Let β be such that $C_r(p, p_k, \beta) = C'(p, q, p_k)$

`Step 2.4:` Store β as $\beta_r(p, p_k)$

`Step 2.5:` Set $B = \{p, p_0, \ldots, pred_B(q), q, p_k\}$.

`End of Algorithm Right` β`-values`

Lemma 8 `Algorithm Right` β`-values` *solves the right β-value problem in $O(n)$ time.*

Proof: The correctness of `Algorithm Right` β`-values` follows from the following observations. Initially the set B is an increasing set. Lemma 5 shows that in `Step 2.2` we find point q with the largest index for which $C_r(p, p_k, \beta_r(p, p_k)) = C'(p, q, p_k)$. Lemma 6 shows that after `Step 2.3` set $B = \{p, p_0, \ldots, pred_B(q), q, p_k\}$ is increasing. So the points $\{p_0, \ldots, pred_B(q)\}$ do not lie inside $C'(p, q, p_k)$. It follows that the β value computed in `Step 2.3` is equal to $\beta_r(p, p_k)$. Moreover, the points in B following q are not relevant in subsequent iterations, as it is shown in Lemma 7. This implies the correctness of the algorithm. Finally, since each point is added to and removed from B at most once, the amortized cost of `Step 2` is $O(n)$. $\square$

The next lemma shows how to compute the right β-values for all pairs of points in arbitrary point sets.

Lemma 9 *Let P be a set of n points in the plane. There exists an algorithm that computes the right β-value for all arcs spanned by the set P in optimal $O(n^2)$ time.*

Proof: The set of right β-values is computed as follows. We first construct the Delaunay triangulation of P [13]. Let (a, b, c) be a triangle in this triangulation. Without loss of generality assume that (a, b, c) is the clockwise order of these points around the triangle. Since $C(a, b, c)$ contains no points from P in it interior, we can immediately compute $\beta_r(a, b)$, $\beta_r(b, c)$ and $\beta_r(c, a)$. Secondly we radially sort $P - \{p\}$ around p for each point p of P [18]. Let $\{p_1, p_2, \ldots, p_m\}$ be the set of points in $C_r(a, b, \infty) \cap C_r(b, c, \infty)$, radially sorted in counterclockwise order around b. We execute `Algorithm Right` β`-values` to obtain $\beta_r(b, p_i)$ for $1 \leq i \leq m$. Repeating this for all three corners of all Delaunay triangles gives us the $\beta_r(u, v)$ for all arcs (u, v). The correctness of the algorithm follows from the correctness of `Algorithm Right` β`-values` (Lemma 8).

Concerning the time complexity of our algorithm, we observe that computing the Delaunay triangulation of P and the radially sorted lists can be done in $O(n^2)$ time; since `Algorithm Right` β`-values` is executed $O(n)$ times and since each execution requires $O(n)$ time by Lemma 8, it follows that the computational cost of the algorithm is $O(n^2)$. Since there are n^2 pairs of points there are n^2 right β-values to compute and the time complexity of the algorithm is asymptotically optimal. $\square$

The results above easily imply the main result of this section.

Theorem 6 *Let P be a set of n points in the plane. There exists an algorithm that computes the circle-based β-spectrum of P in $O(n^2)$ time.*

Proof: For each arc (u, v) we compute $\beta_r(u, v)$ by using the algorithm described in Lemma 9. The β spectrum is easily computed given the $\beta_r(u, v)$ values since $\beta(u, v)$ is the minimum of $\beta_r(u, v)$ and $\beta_r(v, u)$. The bound on the time complexity is a consequence of Lemma 9. $\square$

The next theorem shows how our algorithms of Section 4 can be extended so to efficiently handle the computation the lune-based β-spectrum.

Theorem 7 *Let P be a set of n points in the plane. There exists an algorithm that computes the lune-based β-spectrum of P in $O(n^2)$ time.*

Proof: First we compute the circle-based β-values and keep only those for which $0 \leq \beta \leq 1$. We observe that pairs of points whose lune-based β values are larger than 1 are adjacent in the Delaunay triangulation of P, so there are at most $O(n)$ of these pairs. We can compute the lune-based β-values for any such pair in $O(n)$ time by examining all remaining $n-2$ points.

The overall time complexity is a consequence of Theorem 6, and of the fact that computing a Delaunay triangulation requires $O(n \log n)$ time. □

5 Conclusions and Open Problems

In this paper new algorithms for computing proximity graphs, both in the fixed and in the variable proximity scenario were presented. In the fixed proximity scenario, we showed a simple unifying approach for computing lune-based β skeletons, circle-based β-skeletons, and $[\gamma_0, \gamma_1]$-graphs in $O(n^2)$ time. This approach can be used to devise a numerically robust algorithm that computes the Gabriel graph of a set with optimal degree 2. Regarding the variable proximity scenario, we presented an optimal $O(n^2)$ algorithm for computing the circle-based and the lune-based β-spectrum of a point set.

The research area concerning the design of efficient algorithms for proximity graphs is rich of open problems. We conclude the paper by listing two of those that in our opinion are among the most interesting.

1. It is well-known that for values of β such that $\beta \geq 1$, (closed) lune-based β-skeletons are planar graphs and optimal $O(n \log n)$-time algorithms are known for values of β such that $1 \leq \beta \leq 2$ [14]. Interestingly, no $o(n^2)$-time algorithms are known for values of β such that $2 < \beta \leq \infty$.

2. Devise an algorithm for the computation of Gabriel graph that requires $o(n^2)$ time and optimal degree 2.

Acknowledgments

We are most grateful to Guenter Rote for his remarks that led us to refine the analysis of the time complexity of `Algorithm Right` β`-value`.

References

[1] R. Beirouti and J. Snoeyink. Implementations of the LMT heuristic for minimum weight triangulation. In *ACM Symposium on Computational Geometry*, pp. 96–105, 1998.

[2] J.-D. Boissonnat and F. P. Preparata. Robust plane sweep for intersecting segments. *SIAM J. Comput.*, 29(5) 1401–1421, 2000.

[3] J.-D. Boissonnat and J. Snoeyink. Efficient algorithms for line and curve segment intersection using restricted predicates. *Comput. Geom. Theory Appl.*, 16(1) 35–52, 2000.

[4] T.M. Chan. Reporting curve segment intersections using restricted predicates. *Comput. Geom. Theory Appl.*, 16(4) 245–256, 2000.

[5] F. D'Amore, P.G. Franciosa, and G. Liotta. A robust region approach to the computation of geometric graphs. In *Algorithms–ESA '98*, volume 1461 of *Lecture Notes Comput. Sci.*, pp. 175–186, 1998.

[6] H. Edelsbrunner and L. J. Guibas. Topologically sweeping an arrangement. *J. Comput. Syst. Sci.*, 38 165–194, 1989. Corrigendum in 42 249–251, 1991.

[7] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory*, IT-29 551–559, 1983.

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

[9] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, 4(2) 74–123, 1985.

[10] J. W. Jaromczyk and G. T. Toussaint. Relative neighborhood graphs and their relatives. *Proc. IEEE*, 80(9) 1502–1517, 1992.

[11] J.W. Jaromczyk, M. Kowaluk, and F. Yao. An optimal algorithm for constructing β-skeletons in l_p metric. manuscript, 1989.

[12] M. Karasick, D. Lieber, and L. R. Nackman. Efficient Delaunay triangulations using rational arithmetic. *ACM Trans. Graph.*, 10(1) 71–91, 1991.

[13] D. G. Kirkpatrick and J. D. Radke. A framework for computational morphology. In G. T. Toussaint, editor, *Computational Geometry*, pp. 217–248. North-Holland, Amsterdam, Netherlands, 1985.

[14] A. Lingas. A linear-time construction of the relative neighborhood graph from the Delaunay triangulation. *Comput. Geom. Theory Appl.*, 4 199–208, 1994.

[15] G. Liotta. A note on checking proximity graphs. In *Proceeding 10th Canadian Conference on Computational Geometry*, pp. 106–107, 1998.

[16] G. Liotta, F. P. Preparata, and R. Tamassia. Robust proximity queries: An illustration of degree-driven algo rithm design. *SIAM J. Comput.*, 28(3) 864–889, 1998.

[17] D. W. Matula and R. R. Sokal. Properties of Gabriel graphs relevant to geographic variation research and clustering of points in the plane. *Geogr. Anal.*, 12(3) 205–222, 1980.

[18] M. H. Overmars and Emo Welzl. New methods for computing visibility graphs. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pp. 164–171, 1988.

[19] J. D. Radke. On the shape of a set of points. In G. T. Toussaint, editor, *Computational Morphology*, pp. 105–136. North-Holland, Amsterdam, Netherlands, 1988.

[20] S. V. Rao and A. Mukhopadhyay. Fast algorithm for computing β-skeletons and their relatives. In *Proc. 8th Annual Int. Symp. on Algorithms and Computation*, volume 1350 of *Lecture Notes Comput. Sci.*, pp. 374–383, 1997.

[21] S. V. Rao and A. Mukhopadhyay. Efficient algorithms for computing the β-spectrum. In *Proc. 12th Canad. Conf. Comput. Geom.*, pp. 91–97, 2000.

[22] K. J. Supowit. The relative neighborhood graph with an application to minimum spanning trees. *J. ACM*, 30(3) 428–448, 1983.

[23] R. C. Veltkamp. The γ-neighborhood graph. *Comput. Geom. Theory Appl.*, 1(4) 227–246, 1992.