On Reverse Shortest Paths in Geometric Proximity Graphs

Pankaj K. Agarwal
Department of Computer Science, Duke University, Durham NC, USA

Matthew J. Katz
Department of Computer Science, Ben-Gurion University of the Negev, Beer Sheva, Israel

Micha Sharir
School of Computer Science, Tel Aviv University, Tel Aviv, Israel

Abstract

Let S be a set of n geometric objects of constant complexity (e.g., points, line segments, disks, ellipses) in $\mathbb{R}^2$, and let $\varrho : S \times S \to \mathbb{R}_{\geq 0}$ be a *distance function* on S. For a parameter $r \geq 0$, we define the *proximity graph* $G(r) = (S, E)$ where $E = \{(e_1, e_2) \in S \times S \mid e_1 \neq e_2,\ \varrho(e_1, e_2) \leq r\}$. Given S, $s, t \in S$, and an integer $k \geq 1$, the *reverse-shortest-path* (RSP) problem asks for computing the smallest value $r^* \geq 0$ such that $G(r^*)$ contains a path from s to t of length at most k.

In this paper we present a general randomized technique that solves the RSP problem efficiently for a large family of geometric objects and distance functions. Using standard, and sometimes more involved, semi-algebraic range-searching techniques, we first give an efficient algorithm for the decision problem, namely, given a value $r \geq 0$, determine whether $G(r)$ contains a path from s to t of length at most k. Next, we adapt our decision algorithm and combine it with a random-sampling method to compute r^*, by efficiently performing a binary search over an implicit set of $O(n^2)$ candidate values that contains r^*.

We illustrate the versatility of our general technique by applying it to a variety of geometric proximity graphs. For example, we obtain (i) an $O^*(n^{4/3})$ expected-time randomized algorithm (where $O^*(\cdot)$ hides polylog(n) factors) for the case where S is a set of pairwise-disjoint line segments in $\mathbb{R}^2$ and $\varrho(e_1, e_2) = \min_{x \in e_1, y \in e_2} \|x - y\|$ (where $\|\cdot\|$ is the Euclidean distance), and (ii) an $O^*(n + m^{4/3})$ expected-time randomized algorithm for the case where S is a set of m points lying on an x-monotone polygonal chain T with n vertices, and $\varrho(p, q)$, for $p, q \in S$, is the smallest value h such that the points $p' := p + (0, h)$ and $q' := q + (0, h)$ are visible to each other, i.e., all points on the segment $p'q'$ lie above or on the polygonal chain T.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases Geometric optimization, proximity graphs, semi-algebraic range searching, reverse shortest path

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2022.42

Funding *Pankaj K. Agarwal*: Partially supported by NSF grants IIS-1814493, CCF-2007556, and CCF-2223870.
Matthew J. Katz: Partially supported by Grant 2019715/CCF-20-08551 from the US-Israel Binational Science Foundation/US National Science Foundation.
Micha Sharir: Partially supported by Grant 260/18 from the Israel Science Foundation.

1 Introduction

Let S be a set of n geometric objects of constant complexity (e.g., points, line segments, disks, ellipses) in $\mathbb{R}^2$, and let $\varrho : S \times S \to \mathbb{R}_{\geq 0}$ be a *distance function* on S. For a parameter $r \geq 0$, we define the *proximity graph* $G(r) = (S, E)$, where $E = \{(e_1, e_2) \in S \times S \mid e_1 \neq e_2,\ \varrho(e_1, e_2) \leq r\}$. If S is a set of n points in $\mathbb{R}^2$ and $\varrho(\cdot, \cdot)$ is the Euclidean metric, then

$G(r)$ is the well-studied *unit-disk* graph [15] (where the "unit" here is $r/2$). Here we study proximity graphs in considerably more general settings. For example, let $S = \{e_1, \ldots, e_n\}$ be a set of n pairwise-disjoint segments in $\mathbb{R}^2$. For a pair of segments, $e_1, e_2 \in S$, we define $\varrho(e_1, e_2) = \min_{x \in e_1, y \in e_2} \|x - y\|$, where $\|\cdot\|$ denotes the Euclidean norm. Equivalently, we can define $\varrho(e_i, e_j)$ as follows: For a value $r \geq 0$, let $B(r)$ be the disk of radius r centered at the origin, and let $K_i(r) \coloneqq e_i \oplus B(r) = \{x + y \mid x \in e_i, y \in B(r)\}$ be the Minkowski sum of e_i and $B(r)$. Then $\varrho(e_1, e_2) = \min\{r \mid K_i(r/2) \cap K_j(r/2) \neq \emptyset\}$. See Figure 1 for an illustration. The *segment-proximity graph* $G(r)$ (over S) has an edge between two segments if they are within distance r of each other. See below for more examples of geometric proximity graphs, and note that the above reformulation allows us to interpret proximity graphs as intersection graphs (the unit-disk example is another instance). We note that we do not require $\varrho(\cdot, \cdot)$ to be symmetric. If ϱ is not symmetric, then $G(r)$ is a directed graph in which a directed edge $e_1 \to e_2 \in E$ if $\varrho(e_1, e_2) \leq r$. See Section 1.2 below for an example.

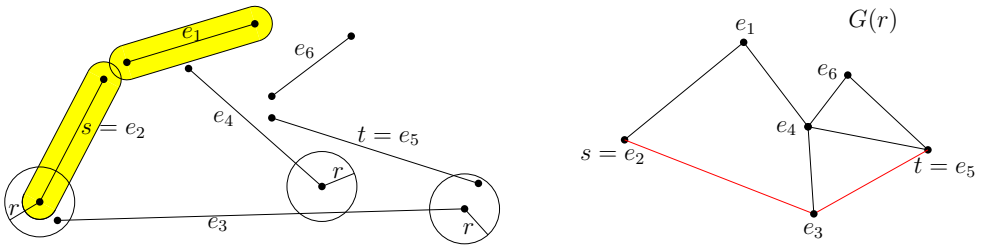

Figure 1 The proximity graph of segments. Left: A possible input consisting of 6 segments, including segments s and t, and a value $r > 0$. The Minkowski sums of e_1 and e_2 with a disk of radius $r/2$ are highlighted. Right: The corresponding proximity graph $G(r)$, where the red edges are those that appear in the shortest path from s to t.

Given S, $s, t \in S$, and an integer $k \geq 1$, the *reverse-shortest-path* (RSP) problem asks for computing the smallest value $r^* \geq 0$ such that $G(r^*)$ contains a path from s to t of length at most k. There are $O(n^2)$ *critical values* of r at which $G(r)$ changes, and our goal is to find the smallest of these critical values for which the proximity graph has the desired property. The RSP problem arises in many applications. For instance, we wish to determine the minimum transmission range for the sensors in a sensor network, such that there is a k-hop transmission path between two given sensors s and t. In this paper we present a general randomized technique for the RSP problem in geometric proximity (and intersection) graphs and apply it to a variety of such graphs.

1.1 Related work

There has been extensive work on understanding the combinatorial structure of geometric proximity graphs (e.g., realizibility of proximity graphs), as well as on developing improved algorithms for a wide range of problems (e.g., shortest paths, vertex covers, independent sets, matchings) on such graphs. Because of numerous applications, to communication networks and other topics, the most widely studied proximity graph is perhaps the unit-disk graph, defined above; see [5, 8, 15, 17, 19] and references therein for a sample of known results on unit-disk graphs. Although a unit-disk graph can have $\Theta(n^2)$ edges, near-linear or subquadratic algorithms are known for many reachability or proximity problems in unit disk graphs, by exploiting the underlying geometry. For example, an $O(n \log n)$ algorithm is known for performing BFS or DFS in unit-disk graphs [10, 12], an $O(n \log^2 n)$ algorithm

for computing shortest paths from a single source in a weighted unit-disk graph (where the weight of an edge (p, q) is $\|p - q\|$ if $\|p - q\| \leq 1$ and ∞ otherwise) [27] (see also [10, 26]), a slightly subquadratic algorithm for computing the diameter of such graphs exactly [12], $O^*(n)$ algorithms[1] for computing the diameter approximately [14] and for computing a spanner [14, 20]. Subquadratic algorithms are known for reachability problems in some other geometric proximity graphs as well, e.g., $O^*(n)$ and $O^*(n^{4/3})$ algorithms for determining whether there exists a path of length at most k between two nodes in disk-intersection or segment-intersection graphs, respectively [6, 13]. Fast algorithms have been developed for computing all pair shortest paths in geometric intersection graphs [13].

In principle, algorithms for geometric proximity graphs could be plugged into the so-called parametric search technique to obtain a subquadratic algorithm for the RSP problem in a geometric proximity graph. However, the difficulty with this approach is that an efficient implementation of the parametric-search technique requires a parallel algorithm for the so-called decision procedure [25], which is not always known. For example, many of the above algorithms use BFS on the resulting graph, but the known BFS algorithms are inherently sequential. Cabello and Jejčič [10] observed that an $O^*(n^{4/3})$ algorithm can be obtained for the RSP problem in a unit-disk graph by using a distance-selection algorithm (given a set S of points in $\mathbb{R}^2$ and an integer $k \geq 1$, return the kth smallest pairwise distance in S; see [3, 22]) to perform a binary search on the $O(n^2)$ pairwise-distances between the input points, and use an efficient BFS algorithm [10] at each step. Wang and Zhao [28] (see also [29]) improved the running time to $O^*(n^{5/4})$ by observing that some of the steps of the BFS algorithm [12] in a unit-disk graph can be parallelized, and they combine parametric search with the distance-selection algorithm. This bound was further improved by Katz and Sharir [21] to $O^*(n^{6/5})$ randomized expected time. We are not aware of any known results on the RSP problem for more general geometric proximity graphs.

We conclude this discussion by noting that the RSP problem has been studied in more general settings, where, given a weighted graph $G = (V, E)$, a pair of nodes $s, t \in V$, and a real parameter W, the goal is to minimize the edge weights as much as possible (subject to various constraints and penalties) so that there is a path in G from s to t of weight at most W. This problem is known to be NP-complete and approximation algorithms are known for a few special cases; see [9, 16, 30] and references therein.

1.2 Our results

There are two main contributions of this paper: First, we describe a general technique for solving the RSP problem in geometric proximity graphs in a fairly general setting. The running time of the algorithm depends on the complexity of the input objects and on the distance function, but it is always subquadratic, even though the proximity graph $G(r)$ may have quadratic size. In all cases described in the paper, this technique yields faster algorithms than those that are based on parametric search because, as mentioned above, the decision algorithms in these cases are hard to parallelize. Second, we consider a wide range of proximity graphs, and develop efficient decision procedures for each case, which are needed to apply our general technique to these instances, and which exploit the geometry of the underlying setup, often in a rather nontrivial manner.

[1] As in the abstract, the $O^*(\cdot)$ notation hides polylog(n) factors.

An overview of our technique. Let S and ϱ be as above. For a pair $e_i, e_j \in S$, we define a predicate $\Pi(e_i, e_j; r)$ that is true if and only if $\varrho(e_i, e_j) \leq r$. Assuming that the objects in S are semi-algebraic sets of constant complexity and that ϱ is also a semi-algebraic function of constant complexity, $\Pi(e_i, e_j; r)$ is a semi-algebraic predicate of constant complexity.[2] Therefore, (e_i, e_j) is an edge in $G(r)$ if and only if $\Pi(e_i, e_j; r)$ holds. Since $\Pi(\cdot)$ is a semi-algebraic predicate of constant complexity, we can efficiently compute a compact representation of $G(r)$, for any given r, as the union of edge-disjoint bipartite cliques (bicliques), with a small overall size of their vertex sets, using either standard halfspace range searching techniques (see [1]), in simpler situations, or (by now standard) semi-algebraic range-searching techniques [2, 4] for more complex setups. With such a compact representation of $G(r)$ at our disposal, the existence of a path from s to t in $G(r)$ of length at most k can be tested in time proportional to the total size of the vertex sets of these graphs by a careful and efficient implementation of BFS on the compact representation of $G(r)$, as in [6]. In other words, this technique yields an efficient decision procedure for the RSP problem, in which r is specified and we want to determine whether $G(r)$ has a path from s to t of length at most k. We describe, in Section 2, the range-searching machinery as well as the efficient BFS implementation on the compact representation of $G(r)$ as the union of bicliques, and show that the expected running time is $O^*(n^{4/3})$.

Next, we obtain a fast optimization algorithm by combining a variant of the decision algorithm with a randomization technique similar to that in [11, 18, 23]. In particular, the technique for computing a compact representation of $G(r)$ is also suitable for computing a compact representation of the graph $G(r, r')$, for $r \leq r'$, in which there is an edge between e_i and e_j if and only if $\varrho(e_i, e_j) \in (r, r']$. By defining the predicate $\Pi(e_1, e_2; r, r')$ to be true if and only if $\varrho(e_i, e_j) \in (r, r']$, we can again use the semi-algebraic range-searching techniques to count the number[3] of *critical values* that lie in the range $(r, r']$. Furthermore, as in [23], the counting procedure is adapted to obtain an efficient randomized procedure for choosing an approximate median of the critical values in a given range $(r, r']$. The expected cost of this procedure is the same (within a polylog(n) factor) as that of the decision procedure. Finally, with this tool available, we solve the optimization problem by a binary search through the critical values of r, using the selection procedure just mentioned to select values for the search, and the decision procedure to guide the search. The overall expected running time for the RSP problem in a segment-proximity graph is $O^*(n^{4/3})$. Unlike some previous application of this randomization techniques, e.g., [23], this approach significantly improves the asymptotic running time of the algorithm in our applications, over what one could have obtained using parametric search.

We remark here that the application of range searching in many cases here is non-trivial and requires new ideas, so that we can use multi-level data structures [1] in as low dimension as possible and obtain better bounds than what we would obtain by a naïve application. For example, a naïve application of semi-algebraic range searching in the running example of segment-proximity graphs would result in a bound of $O^*(n^{8/5})$, as the number of parameters needed to specify an input segment (i.e., its two endpoints) is four, and even simplex (batched) range searching in $\mathbb{R}^4$ results in the aforementioned bound. However, by constructing a

[2] Roughly speaking, a *semi-algebraic set* in $\mathbb{R}^d$ is the set of points in $\mathbb{R}^d$ satisfying a Boolean predicate over a set of polynomial inequalities; the complexity of the predicate and of the set is defined in terms of the number of polynomials involved and their maximum degree. A real-valued function is called semi-algebraic if its graph is a semi-algebraic set. See [7] for details.

[3] In many cases, we define the set of critical values to be a superset of $\{\varrho(e_i, e_j) \mid e_i, e_j \in S\}$, but this does not affect the general approach.

multi-level structure carefully, in which the input objects at each level (features of the actual input objects) can be specified by only two parameters, we succeed in obtaining the bound of $O^*(n^{4/3})$; see Section 2.

Efficient solutions to various specific proximity graphs. We illustrate the versatility of our technique by applying it to a wide range of geometric proximity graphs, as listed below.

Communication graphs of directional antennas. Let $S = \{p_1, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^2$, where each point p_i has a direction $u_i \in \mathbb{S}^1$ associated with it, and let $\delta > 0$ be a *range* parameter. For a parameter $\alpha \in [0, 2\pi]$, let $A_i(\alpha)$ denote the *directional antenna* of range δ located at p_i whose symmetry axis is u_i and which has an opening angle α, i.e., $A_i(\alpha) = \{x \in \mathbb{R}^2 \mid \|x - p_i\| \leq \delta \ \wedge \ \langle u_i, x - p_i\rangle \geq \cos(\alpha/2)\}$. (The case where each antenna has its own range will also be considered.) We consider both asymmetric and symmetric distance functions: For a pair of points $p_i, p_j \in S$ with $\|p_i - p_j\| \leq \delta$, we define $\varrho(p_i, p_j) = \min\{\alpha \in [0, 2\pi] \mid p_j \in A_i(\alpha)\}$ for the asymmetric version, and $\varrho(p_i, p_j) = \min\{\alpha \in [0, 2\pi] \mid p_j \in A_i(\alpha) \ \wedge \ p_i \in A_j(\alpha)\}$ for the symmetric version. We set $\varrho(p_i, p_j) = \infty$ if $\|p_i - p_j\| > \delta$ in both cases. See Figure 2 for an illustration. The *communication graph* $G(\alpha)$ is a directed graph for the asymmetric version and undirected for the symmetric version.

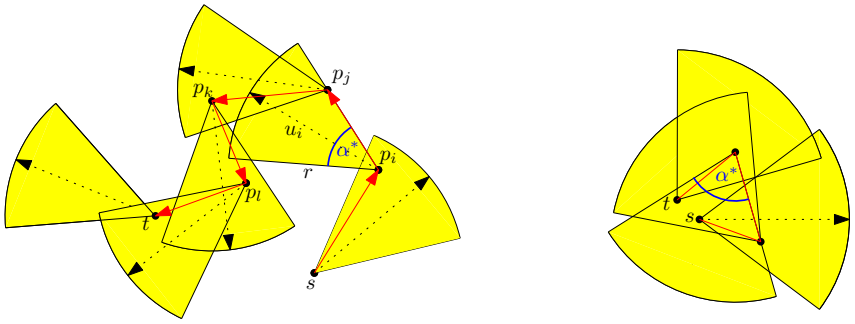

Figure 2 The directional antennas problem. Left: The asymmetric version with $k = 5$. Right: The symmetric version with $k = 3$.

We present (in Section 3.1) a randomized algorithm, with $O^*(n^{4/3})$ expected running time, for the RSP problem in the communication graph of directional antennas, for both symmetric and asymmetric versions. We then modify the algorithm to accommodate the case where each antenna has its own range. The expected running time of this latter algorithm is $O^*(n^{7/5})$.

Polyline-proximity graphs. The polyline-proximity graph is a generalization of the segment-proximity graph. The input consists of a set $\mathcal{T}$ of n pairwise-disjoint polygonal chains (representing roads, say), each of size (i.e., number of vertices) at most some constant l. For a pair $T_i, T_j \in \mathcal{T}$, we define $\varrho(T_i, T_j) = \min_{p \in T_i, q \in T_j} \|p - q\|$. As for segment-proximity graphs, an equivalent formulation is: For a value $r \geq 0$, let $K_i(r) := T_i \oplus B(r)$. Then $\varrho(T_i, T_j) = \min\{r \geq 0 \mid K_i(r/2) \cap K_j(r/2) \neq \emptyset\}$. See Figure 3 for an illustration. We present (in Section 3) a randomized algorithm, with $O^*(n^{4/3})$ expected time, for the RSP problem in polyline-proximity graphs.

Graphs of growing segments. Let $S = \{p_1, p_2, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^2$, so that each point p_i is associated with a direction $u_i \in \mathbb{S}^1$. For $r > 0$, let $e_i(r) := p_i + ru_i$ denote the segment of length r that emanates from p_i in direction u_i. (We can also consider cases

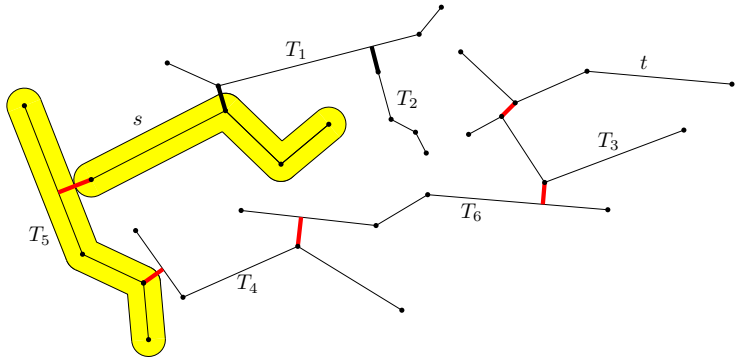

Figure 3 The proximity graph of polylines. In this example, $l = 4$, $k = 5$, and r^* is the distance between s and T_5 (the Minkowski sums of s and T_5, respectively, with a disk of radius $r/2$ are highlighted). The edges of the graph are drawn as either black or red thick segments, where the red ones are those that appear in the path (of length 5) between s and t.

where each $e_i(r)$ grows at a different scale, namely $e_i(r) := p_i + \lambda_i r u_i$, for suitable scalars λ_i.) We now define $\varrho(p_i, p_j) = \min_r\{r \geq 0 \mid e_i(r) \cap e_j(r) \neq \emptyset\}$. See Figure 4 for an illustration of the special case where all the points in P are on the x-axis. We present (in Section 3.2) a randomized algorithm, with $O^*(n^{4/3})$ expected time, for the RSP problem in graphs of growing segments.

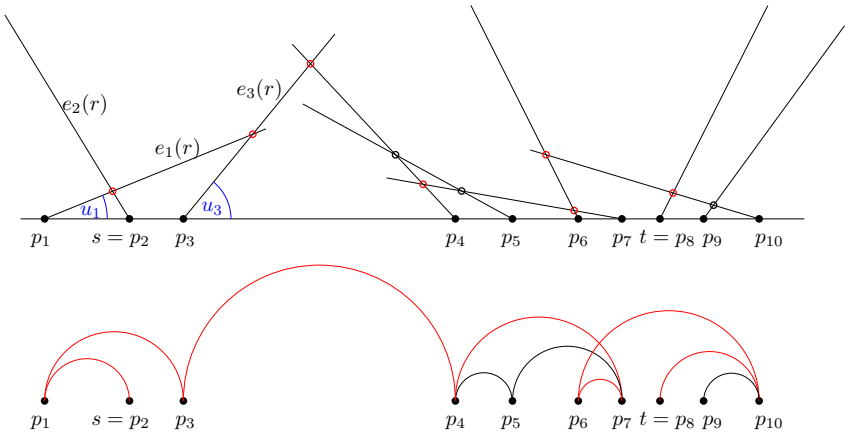

Figure 4 The growing segments problem for an instance where the points are on a line. Top: A possible input consisting of 10 points, including points s and t, and their associated directions, with some common growth value $r > 0$. Bottom: The corresponding graph $G(r)$. The edges in red are those that appear in the shortest path from s to t, which is $(s, p_1, p_3, p_4, p_7, p_6, p_{10}, t)$. These edges correspond to the intersection points, marked in red, between segments. In this example $r^* < r$; more precisely, r^* is the length of the segment with endpoints p_1 and $e_1(r) \cap e_3(r)$.

Variants of growing-segment and segment-proximity graphs. Let $S = \{p_1, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^2$. We generalize the growing-segment graph by considering other shapes (e.g., ellipses) that can be grown around each site p_i. In the simplest version, which we will follow, we assume that each shape has only one degree of freedom of growth. For example, when growing ellipses, we may assume that the ellipse grown at a site p_i is centered at p_i, has fixed directions θ_i and $\theta_i + \frac{\pi}{2}$ of its axes, and has a fixed ratio λ_i between the lengths

of its major and minor axes. We denote this ellipse as $E_i(r)$, where r, the single degree of freedom of growth, is half the length of the major axis. For a pair of points $p_i, p_j \in S$, we now define $\varrho(p_i, p_j) = \min\{r \geq 0 \mid E_i(r) \cap E_j(r) \neq \emptyset\}$. See Figure 5(a) for an illustration.

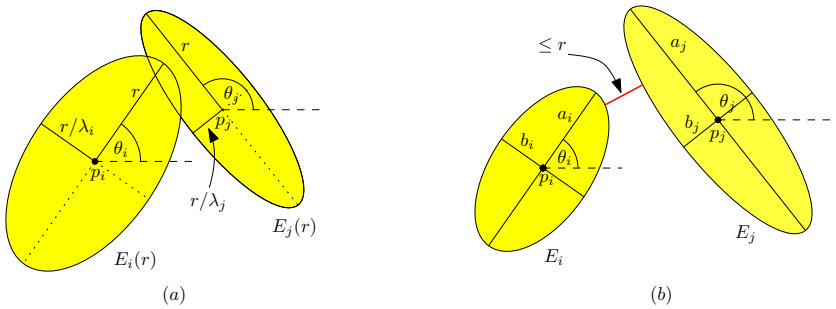

Figure 5 (a) The growing ellipses problem. The ellipse $E_i(r)$ is centered at p_i, the direction of its major axis is θ_i and its length is $2r$; the direction of the minor axis is $\theta_i + \frac{\pi}{2}$ and its length is $2r/\lambda_i$. Since $E_i(r) \cap E_j(r) \neq \emptyset$, (p_i, p_j) is an edge of $G(r)$. (b) The ellipse proximity problem. The length of E_i's major and minor axes are a_i and b_i. Since the distance between E_i and E_j is less than or equal to r, (E_i, E_j) is an edge of $G(r)$.

We present, in Section 3, a randomized algorithm, with $O^*(n^{8/5})$ expected time, for the RSP problem in the growing-ellipse graph. It exploits the fact that each input ellipse has four degrees of freedom (except for the growth parameter r). Here the predicate that determines whether $E_i(r) \cap E_j(r) \neq \emptyset$ is a semi-algebraic predicate of constant complexity, and the associated range-searching problem is with semi-algebraic ranges in four dimensions. The running time here is worse than those in the preceding problems because the associated range-searching problem is in higher dimensions. Currently, we do not see how to reduce the dimension, as we could in the previous problems, and leave this as an open challenge.

Similarly, we can generalize the segment-proximity graph by replacing segments with other shapes. For example, we can have a set $S = \{E_1, \ldots, E_n\}$ of n pairwise-disjoint ellipses in $\mathbb{R}^2$, and we now define $\varrho(E_i, E_j) = \min_{p \in E_i, q \in E_j} \|p - q\|$. See Figure 5(b) for an illustration. As a general ellipse in the plane has five degrees of freedom, our technique solves the RSP problem in the ellipse-proximity graph in $O^*(n^{5/3})$ expected time, by using five-dimensional semi-algebraic range searching data structures. As before, it would be an interesting challenge to improve this bound.

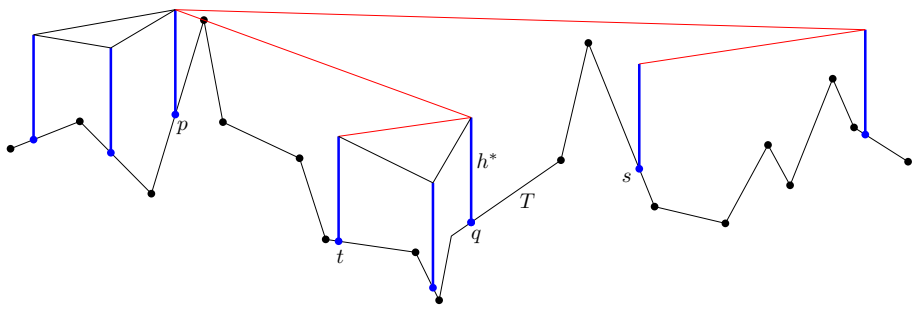

Figure 6 Visibility graph of towers over a terrain. The set Q consists of 8 points, including s and t, and $k = 4$. The height h^* is determined by the pair p, q. The segments between tips of towers correspond to the edges of the visibility graph $V(h^*)$, where the red ones correspond to those that participate in the path, of length 4, between s and t.

Visibility graph of towers over a terrain. Here we face a different setup. We have an x-monotone polygonal line T with n vertices, to which we refer as a 1.5-*dimensional terrain*, and a set Q of m points on T (in general not at its vertices). For a point $q_i \in Q$ and a value $h \geq 0$, let $q_i(h) := q_i + (0, h)$ denote the translate of q_i in the vertical direction by distance h. We can view $q_i(h)$ as the tip of a tower of height h erected on q_i. For a pair $q_i, q_j \in Q$, we now define $\varrho(q_i, q_j)$ to be the minimum value h for which the segment $q_i(h)q_j(h)$ is *visible*, i.e., all the points on the segment lie above or on T. We refer to $G(h)$ as the *visibility graph of towers (of height h) over a terrain*. See Figure 6 for an illustration.

We present (in Section 3.3) a randomized algorithm, with $O^*(n + m^{4/3})$ expected time, for the RSP problem in visibility graphs of towers. As already remarked, this problem is more general than the preceding problems, in that each critical value of h does not depend solely on a pair of points $p, q \in S$ but also on the entire portion of T between p and q. This makes the selection procedure more involved, but we can still make it efficient using a more careful analysis, based on an adaptation of a technique of Agarwal-Varadarajan [6].

2 RSP in Segment-Proximity Graphs

In this section we describe the details of our technique by illustrating it for segment-proximity graphs. Let $S = \{e_1, e_2, \ldots, e_n\}$ be a set of n pairwise-disjoint segments in the plane, let s, t be two segments in S, and let $k \geq 1$ be an integer.

The decision procedure. Consider first the decision problem, in which r is specified and we want to determine whether $r^* \leq r$, i.e., whether $G(r)$ contains a path from s to t of length at most k. We present an algorithm that solves the decision problem in $O^*(n^{4/3})$ time. The solution begins by representing $G(r)$ as the union of bipartite cliques (which are not necessarily edge disjoint). We represent each $e_i \in S$ by the pair of its endpoints, and note that each e_i has four degrees of freedom.

The condition that $\varrho(e_i, e_j) \leq r$ can be expressed as a semi-algebraic predicate of constant complexity, as follows. The distance between two disjoint segments is attained either between two endpoints, one of each segment, or between an endpoint of one segment and the relative interior of the other. We can thus write the condition that a segment $e = uv$ lies at distance at most r from a segment $e' = pq$ as a semi-algebraic predicate $\Pi(e, e'; r)$ which is a disjunction of sub-predicates, each of which is a conjunction of several conditions. The efficiency of the procedure stems from the fact that we can guarantee that each of these sub-conditions involves at most two parameters from the four parameters representing each segment. For example, if the distance between e and e' is attained at their respective endpoints u and p then (i) u and e' lie on different sides of the line ℓ_p^+ that is orthogonal to e' at p, (ii) p and e lie on different sides of the line ℓ_u^+ that is orthogonal to e at u, and (iii) $\varrho(u, p) \leq r$. As another example, if the distance between e and e' is attained at u and a point in the relative interior of e' then (i') u and e' lie on the same side for each of the lines ℓ_p^+ and ℓ_q^+ (the line orthogonal to e' at q), and (ii') $\varrho(u, \ell_{e'}) \leq r$, where $\ell_{e'}$ is the line supporting e'. See Figure 7. The first set of conditions (i)–(iii) are necessary and sufficient for $\varrho(e, e')$ to be attained at p and u (and be at most r), as is easily checked. However, the second set of conditions (i')–(ii') are necessary but not sufficient conditions for $\varrho(e, e')$ to be attained at u and an interior point of e'; see Figure 7(iv). Similar conjunctions arise for all other possible cases. Moreover, whenever one of these conjunctions holds we have $\varrho(e, e') \leq r$.

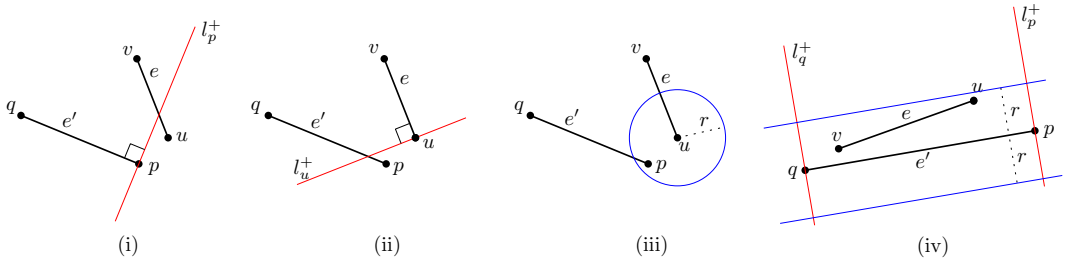

Figure 7 Illustrating conditions (i)–(iii) for the case where the distance between the segments $e = uv$ and $e' = pq$ is attained between their endpoints u and p. (iv) Illustrating conditions (i')–(ii') and the fact that they are not sufficient for $\varrho(e, e')$ to be attained at u and an interior point of e'.

Representing $G(r)$ by bipartite cliques. We turn the problem of constructing $G(r)$ into a batched range searching problem, in which the segments of S serve both as input objects and as queries. The query with a segment e defines the range $Q_e = \{e' \in S \mid \Pi(e, e'; r) \text{ holds}\}$.

We prepare a separate data structure for each of the aforementioned sub-predicates whose disjunction is $\Pi(e, e'; r)$. For example, consider the case where the distance is attained between two endpoints u and p, using the above notation. For sub-condition (i), we prepare a batched halfplane range searching structure, with the endpoints u as input and the halfplanes bounded by the lines ℓ_p^+ and not containing the respective edges e' as ranges. The next level handles sub-condition (ii), in a completely analogous and symmetric manner, and the third level enforces sub-condition (iii), in which the input are the points p and the ranges are disks of radius r around the points u. Similar multi-level structures are constructed for the other sub-predicates in the conjunction. For example, when testing for conditions (i')–(ii'), the last level tests whether the distance from endpoint u to the line $\ell(e')$ supporting e' is at most r. This amounts to testing whether u lies in the strip of width $2r$ centered at $\ell(e')$, and can be implemented using two levels of halfspace range queries (or rather point enclosure queries). Using standard results on multi-level range searching structures (see, e.g., [1]), the whole procedure, with n input objects and n query ranges, takes $O^*(n^{4/3})$ time, and produces $G(r)$ as the union of a collection of (not necessarily edge-disjoint) bipartite cliques, so that the sum of the sizes of their vertex sets is also $O^*(n^{4/3})$.

Finding the path by BFS. With this data available, the rest of the path searching algorithm is relatively easy. We run a BFS through $G(r)$, starting from s. A similar algorithm has been given in [6], and we only briefly sketch the details. At each stage of the BFS we iterate over the points in the current layer, and use the bicliques in which they participate to propagate the path to the next layer. Say we reach a graph $A \times B$, and that the current vertex (i.e., segment) v being processed belongs to A. We then add all the unvisited nodes of B to the next layer of the BFS. When we later process a vertex of B, we repeat the above propagation step, swapping A and B, and then discard the graph completely. In applications where the graph $G(r)$ is directed (see Section 3.1 for examples of such problems), the treatment is somewhat different, with obvious modifications. We keep performing these BFS propagation steps until either t is reached, within the first k layers of the BFS, or we have created k layers without reaching t, or we run out of graphs to process. The overall cost of the BFS is proportional to the overall size of the vertex sets of the graphs, which is $O^*(n^{4/3})$. See [6] for further details.

Solving the optimization problem. The optimization procedure is a variant, or rather an extension, of the distance-selection mechanism of [3], with one notable difference that we replace the parametric search technique used in [3] by a simpler, and as it turns out more efficient, random sampling approach [23]. Our algorithm is based on a procedure, in which, given a threshold value r_0, we want to count the number of critical values that are smaller than or equal to r_0. A value r is *critical* if there exist a pair of segments e_i, e_j in S such that one of the sub-predicates in the disjunction forming $\Pi(e_i, e_j; r)$ holds and its associated critical value is exactly r. Note that the set of critical values is a superset of the set of actual distances between the pairs of segments in S (see, for instance, Figure 7(iv)). This enlargement of the set of critical values is made for technical reasons, to address the fact that the bipartite clique representation of $G(r_0)$ is not edge disjoint; see below for details. (Note that in degenerate configurations the same value r can arise for more than one pair (e_i, e_j), which means that we regard the set of critical values as a multiset.)

Note that we do not count the number of edges of $G(r_0)$, but rather the sum of the weights of these edges, where the *weight* of an edge (e_i, e_j) is the number of satisfied sub-predicates in the disjunction forming $\Pi(e_i, e_j; r_0)$, that is, the number of subgraphs it appears in. We denote this sum of weights as $\mu(G(r_0))$.

We use the quantity $\mu(G(r_0))$ because it is straightforward to compute from the bipartite clique decomposition of $G(r_0)$, in time proportional to the overall size of the vertex sets of the bipartite graphs, that is, in $O^*(n^{4/3})$ time. The quantity $\mu(G(r))$ is (weakly) monotone increasing in r, as easily follows from its definition. As we will see shortly, using $\mu(G(r_0))$ simplifies the optimization procedure and does not affect its performance in any significant manner.

Let $\hat{r}$ be some upper bound on the critical values, over all pairs (e_i, e_j) of segments in S. It is easy to compute $\hat{r}$ in linear time by, e.g., computing the smallest enclosing axis-parallel square of the segments in S. We begin by computing the bipartite clique representation of the graph $G(\hat{r})$. Next, by running a BFS in $G(\hat{r})$, starting from s, as described above, we determine whether $G(\hat{r})$ contains a path between s and t of length $\leq k$. If it does not, then we output that r^* does not exist and stop. Assume therefore, that $G(\hat{r})$ does contain an s-t path of the desired length.

At this point, we know that r^* is in the range $(0, \hat{r}]$, and we narrow down the range using binary search in the set of critical values, where comparisons of the form "$r^* \leq r$?" are resolved by a call to the decision procedure with r. To run the binary search, we define the graph $G(r_1, r_2)$, $r_1 \leq r_2$, whose set of vertices is S and there is an edge between e_i and e_j if one or more of the critical values corresponding to them is in the range $(r_1, r_2]$. Notice that $G(\hat{r}) = G(0, \hat{r})$. Given $r_1 \leq r_2$, we can compute a bipartite clique representation of $G(r_1, r_2)$ in $O^*(n^{4/3})$ time, where the overall size of the vertex sets in this representation is also $O^*(n^{4/3})$, by the following modification of the procedure given above. For each of the multi-level structures used by the preceding procedure, we change its bottom level, so that it collects all pairs of segments e, e' for which the corresponding distance (between two specific endpoints or between an endpoint of one segment and the line supporting the other segment) is in $(r_1, r_2]$. For the case where the considered distance is between two endpoints, we apply range searching with annuli, of the fixed radii r_1 and r_2, instead of the disks used earlier. For the case where the considered distance is between an endpoint and a line, we apply range searching with pairs of strips of width $r_2 - r_1$, obtained as the set differences of the corresponding strips of widths $2r_2$ and $2r_1$. This does not affect the asymptotic performance bounds – it only adds levels to the structures.

We note the following properties: (i) A pair of segments (e, e') in S that has already been encountered while computing $G(r_1)$ may appear again, due to a different pair of features on e and e' being at distance in $(r_1, r_2]$. (ii) Nevertheless, the sum $m = \mu(G(r_1, r_2))$ of the number of edges in the biclique representation of $G(r_1, r_2)$ is exactly $\mu(G(r_2)) - \mu(G(r_1))$, as is implied by the construction.

Assume that we already know that r^* is in the range $I = (r_1, r_2]$. To perform the next binary search step, we compute an approximate median r of the critical values in I, such that the number of critical values in each of the intervals $I_1 = (r_1, r]$ and $I_2 = (r, r_2]$ is, say, at most $2m/3$. This can be done by taking a constant-size random sample of critical values in I (see below for details), and returning the median value r of the sample. It is well-known and easy to see that for a sufficiently large (but still constant) sample size, r is an approximate median as desired with probability greater than $1/2$; see, e.g., [11, 18, 23].

We thus check whether r is indeed an approximate median, and repeat the process with a new random sample, for an expected constant number of times, if it is not.

Once we have found an approximate median r, we run the decision procedure at r to determine whether $r^* \leq r$. If the answer is "YES", we continue with the range I_1, and if it is "NO", we continue with the range I_2. When the number $\mu(G(r_1, r_2))$ of the critical values in the current range is, say, $O(n)$, we compute these values explicitly, by modifying our data structures so that they report critical values instead of counting them, and perform a binary search among them to find r^*.

It remains to describe how to compute a random sample of c critical values in the range $I = (r_1, r_2]$. Consider the representation of $G(r_1, r_2)$ by the union of bicliques, enumerated as $A_1 \times B_1, \ldots, A_\nu \times B_\nu$, where each of the sets A_i, B_i is also enumerated in some arbitrary order. This latter enumeration induces a natural lexicographical order of the edges of each $A_i \times B_i$. Recall that $m = \mu(G(r_1, r_2))$ is the sum $\sum_{i=1}^{\nu} |A_i| \cdot |B_i|$. To pick a random critical value, we draw a random number t in $[1, m]$, and find the index j for which $\sum_{i=1}^{j-1} |A_i| \cdot |B_i| < t \leq \sum_{i=1}^{j} |A_i| \cdot |B_i|$. We then find the t'-th edge of $A_j \times B_j$ in lexicographical order, where $t' := t - \sum_{i=1}^{j-1} |A_i| \cdot |B_i|$, using a similar procedure, and return the associated critical value. (Since we focus on a specific biclique, which corresponds to a single sub-predicate in the disjunction forming Π, the critical value is uniquely defined.)

All this leads to the following summary result.

▶ **Theorem 1.** *The RSP problem in the proximity graph of n pairwise-disjoint segments in the plane can be solved in $O^*(n^{4/3})$ time.*

3 RSP in Other Proximity Graphs

The scheme presented in the previous section applies to many other optimization problems of a similar nature. We discuss in this section and in Appendix A several such problems.

The general approach: A high-level overview of the decision procedure. Generally, the decision procedure for determining whether $G(r)$ has the desired property for a given r, has to construct $G(r)$ as a union of bicliques, which reduces to batched range searching with constant-complexity semi-algebraic ranges in some suitable dimension.

Once $G(r)$ is available, testing for the existence of a path of length at most k is done using BFS, as in the preceding section. The same machinery can also be applied to count the number of critical values that are smaller than or equal to r or that lie in a range $(r, r']$.

The performance of these steps depends on the ambient dimension t, or the dimensions ξ, η, of the parametric spaces that represent the features of the input and query objects that participate in any sub-predicate of the predicate $\Pi(e, e'; r)$ that represents the property that

the distance between objects e and e' is $\leq r$. In the symmetric case, where both the primal space (in which the objects e are stored as points) and the dual space (in which the objects e' are stored as points) have the same dimension t, the recently developed machinery for range searching with semi-algebraic sets [2, 4, 24][4] implies that this cost is $O^*(n^{2t/(t+1)})$, where n is the total number of objects. In the asymmetric case, where the primal and dual parametric spaces have different respective dimensions ξ, η, and we have n objects in the former and m in the latter, the more general bound, developed in Appendix B, applies, and this cost is $O^*(m^{\xi(\eta-1)/(\xi\eta-1)}n^{\eta(\xi-1)/(\xi\eta-1)} + m + n)$. Since we use a multi-level structure, the parameters t, ξ, η may differ at different levels; the overall cost is dominated by the cost of the most expensive level (up to the $O^*(\cdot)$ notation). This machinery uses polynomial partitions. It can be replaced by cuttings, when the parametric dimension t, ξ, or η is at most four or when the predicate only contains linear inequalities.

We now proceed to list the other problems mentioned in the introduction, to which our technique can be applied, and discuss the concrete solution of each of them. Due to lack of space, we delegate some of these problems to Appendix A.

3.1 Reverse shortest paths in communication graphs of directional antennas

Recall that here we are given a set P of n points, including two designated points s and t, a range $\delta > 0$, and an integer $k \geq 1$. Moreover, each point p_i is associated with a direction u_i. The problem is to find the smallest angle α^* such that, if we place at each point p_i a *directional antenna* $A_i(\alpha^*)$ of range δ, symmetry axis u_i and opening angle α^*, then there is a path from s to t of length at most k in the induced communication graph $G(\alpha^*)$. (Alternatively, one can fix the opening angle of the antennas and ask to minimize the range. Our technique works for this case as well.) We can consider either the asymmetric version, where $G(\alpha^*)$ is a directed graph over P, whose edges are all the pairs (p_i, p_j) for which $p_j \in A_i(\alpha^*)$, or the symmetric version, where $G(\alpha^*)$ is undirected and its edges are all the pairs (p_i, p_j) for which both $p_j \in A_i(\alpha^*)$ and $p_i \in A_j(\alpha^*)$. See Figure 2 for an illustration. We present an algorithm for (either version of) this problem that runs in $O^*(n^{4/3})$ time.

Consider, for concreteness, the asymmetric version of the problem (the symmetric version is solved by essentially the same technique). The number of parameters needed to represent an antenna is three, two for p_i and one for u_i (δ is treated as a constant, and α is the growth parameter that we want to minimize). For a fixed pair p_i, p_j, we can write the condition that $p_j \in A_i(\alpha)$ as a semi-algebraic predicate $\Pi(p_i, u_i, p_j; \alpha)$ which is a disjunction of two sub-predicates, each of which is a conjunction of several conditions, each involving only two out of the three parameters representing $A_i(\alpha)$. Namely, $p_j \in A_i(\alpha)$ if and only if (i) p_j lies on the "right" side of both the lines supporting the rays u_i^- and u_i^+, which are obtained from u_i by rotating it by an angle of $\alpha/2$ in clockwise and counterclockwise directions, respectively, and (ii) $\varrho(p_i, p_j) \leq \delta$.

We thus construct two range searching structures, one for each of the sub-predicates of $\Pi(p_i, u_i, p_j; \alpha)$, where each structure consists of three levels, similar to the structures constructed for the segment proximity problem. Finally, we perform n queries, one for each point in P, in each of these structures, to obtain the bipartite clique representation of $G(\alpha)$. As in the segment proximity problem, the overall running time is $O^*(n^{4/3})$, which also bounds

[4] The primal-dual combination of the techniques of [2, 24], which is rather non-trivial, has not been explicitly treated in any previous work, as far as we can tell.

the overall size of the vertex sets in the compact representation of $G(\alpha)$. This follows from the property that at each level of the structure, the features of the antennas that participate in that level have only two degrees of freedom.

Once this representation of $G(\alpha)$ is available, testing for the existence of a path of length k between s and t can be done, similar to Section 2, using a careful implementation of BFS on that graph. We note that in the asymmetric version of the problem the graph is directed. This means that when we process a vertex v that participates in some (now directed) bipartite clique $A \times B$, with $v \in A$, we put in the next layer of the BFS only the still unvisited vertices of B, and discard $A \times B$ from further processing, even though some of its A-vertices might still not have been visited. When such a vertex is visited later, the graph is of no use, as all its B-vertices have already been visited. Except for this difference, the BFS and the decision procedure proceed as before. The bound $O^*(n^{4/3})$ also dominates the overall running time of the entire procedure.

One can also consider the case where each antenna has its own range. Informally, this simply means that we add one extra parameter, namely the range, to the tuple of parameters that specify an input antenna, so each antenna is represented now by four parameters. The case can be treated similarly except that the number of parameters that specify an antenna (ignoring the growth parameter α) is now four, instead of three. The only difference is that now, when performing a query for p_i, we use the disk of radius δ_i around p_i at the bottom level of the structure in which the query is performed. In other words, at the bottom level we apply the algorithm for mixed-dimensional batched range searching with $\xi = 2$ (for the input objects p_j) and $\eta = 3$ (for the query objects $A_i(\alpha)$), which, by Equation (1) of Appendix B, takes a total of $O^*(n^{7/5})$ running time, which also dominates the overall running time of the entire procedure. We thus obtain:

▶ **Theorem 2.** *The RSP problem in the communication network of n directional antennas in the plane can be solved in $O^*(n^{4/3})$ time when all the antennas have the same range, and in $O^*(n^{7/5})$ time when each antenna has a different range. This holds for both symmetric and asymmetric versions of the problem.*

3.2 Reverse shortest paths in intersection graphs of growing segments in the plane

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of n points in the plane, so that each point p_i is associated with a direction u_i. For $p_i \in P$, we denote by $e_i(r)$ the segment of length r that emanates from p_i in direction u_i. Let $G(r)$ be the intersection graph of these segments. That is, $G(r)$ is the graph over P, whose set of edges consists of all the pairs (p_i, p_j) for which $e_i(r) \cap e_j(r) \neq \emptyset$. Let s and t be two designated points of P. In the growing segments problem, we wish to find the smallest value r^* of r, such that there is a path in $G(r^*)$ from s to t of length at most some prescribed value k. See Figure 4 for an illustration of the special case where all the points in P are on the x-axis.

We present an algorithm for this problem that runs in $O^*(n^{4/3})$ time.

As in the directed antennas problem, the number of parameters needed to represent an input object is three, two for p_i and one for u_i (where r is the growth parameter that we wish to minimize). For a fixed pair p_i, p_j, we can write the condition that $e_i(r) \cap e_j(r) \neq \emptyset$ as a semi-algebraic predicate $\Pi(p_i, u_i, p_j, u_j; r)$. Concretely, $e_i(r) \cap e_j(r) \neq \emptyset$ if and only if the endpoints of $e_i(r)$ lie on different sides of the line supporting $e_j(r)$, and vice versa. By trying all possible combinations of sides, we can write $\Pi(p_i, u_i, p_j, u_j; r)$ as a disjunction of several sub-predicates, each of which is a conjunction of four conditions, each requiring some

specific endpoint of one segment to lie on some specific side of the line supporting the other segment. Note that each of these sidedness conditions involves only two out of the three parameters representing each of the objects.

We thus construct several range searching structures of an identical nature, one for each combination of sides. Each structure consists of four levels, each of which is a structure for batched halfplane range searching. We perform n queries, one for each point in P, in each of these structures, to obtain the bipartite clique representation of $G(r)$. Omitting the straightforward details, the overall running time is $O^*(n^{4/3})$, which also bounds the overall size of the vertex sets in the compact biclique representation of $G(\alpha)$. This bound also dominates the overall running time of the entire algorithm, and we get:

▶ **Theorem 3.** *The RSP problem for n growing segments in the plane can be solved in $O^*(n^{4/3})$ time.*

3.3 Visibility graph of towers over a terrain

Recall that here we are given an x-monotone polygonal line T (i.e., a 1.5-*dimensional terrain*) with n vertices, a set Q of m points on T (not necessarily vertices), including two designated points s and t, and an integer parameter $1 \leq k < m$. The problem is to find the minimum height h^* such that if we place a tower of height h^* at each point of Q, then there exists a path of length at most k between the tips of the towers at s and t in the *visibility graph* $G(h^*)$ as defined earlier. See Figure 6 for an illustration. We present an algorithm for this problem that runs in $O^*(n + m^{4/3})$ time.

We note that this problem is just one of several problems whose corresponding decision procedure constructs, as a preliminary stage, a compact representation of the visibility graph $G(h)$, as just defined. This task (i.e., computing the representation of $G(h)$) is different than the analogous tasks in the other problems considered in this paper, since the decision whether an edge (q, q') belongs to $G(h)$ does not depend only on q and q' (and on h) but also on the entire portion of T between q and q'. Nevertheless, adapting a technique proposed by Agarwal and Varadarajan [6], we can compute the desired representation of $G(h)$ in $O^*(n + m^{4/3})$ time.

Constructing a compact representation of $G(h)$

We use a divide-and-conquer approach. Enumerate the vertices of T as $p_1, \ldots, p_n$ from left to right. Partition T into two subpaths $T_L = (p_1, p_2, \ldots, p_{\text{med}})$ and $T_R = (p_{\text{med}}, p_{\text{med}+1}, \ldots, p_n)$, where $\text{med} = \lfloor n/2 \rfloor$. Let $Q_L = Q \cap T_L$ and $Q_R = Q \cap T_R$, and put $m_L = |Q_L|$, $m_R = |Q_R|$. We compute recursively a compact representation of $G(h)_L = \{qq' \mid qq' \in G(h), q, q' \in Q_L\}$ and of $G(h)_R = \{qq' \mid qq' \in G(h), q, q' \in Q_R\}$, and face the problem of computing (a compact representation of) the portion of $G(h)$ consisting of edges that connect a point in Q_L with a point in Q_R.

In the variant of the technique of Agarwal and Varadarajan that we use here, we take each point $q \in Q_L$, and find the upper rightward-directed tangent ray $\tau^+(q)$ from $q(h)$, the tip of the tower of height h at q, to the subchain $(p_k, \ldots, p_{\text{med}})$ of T_L, where p_k is the leftmost vertex of T_L to the right of q. Symmetrically, for each point $q' \in Q_R$, we find the upper leftward-directed tangent ray $\tau^-(q')$ from $q'(h)$ to the subchain $(p_{\text{med}}, \ldots, p_{k'})$ of T_R, where $p_{k'}$ is the rightmost vertex of T_R to the left of q'. Then $q(h)$ and $q'(h)$ are mutually visible if and only if $q'(h)$ lies above $\tau^+(q)$ and $q(h)$ lies above $\tau^-(q')$; see Figure 8.

To find the tangents $\tau^+(q)$, for $q \in Q_L$, we construct a balanced binary tree over the vertices of T_L, and construct, at each node v of the tree, the upper convex hull of the vertices of T_L that are stored at the subtree rooted at v. With some care, the overall cost of these

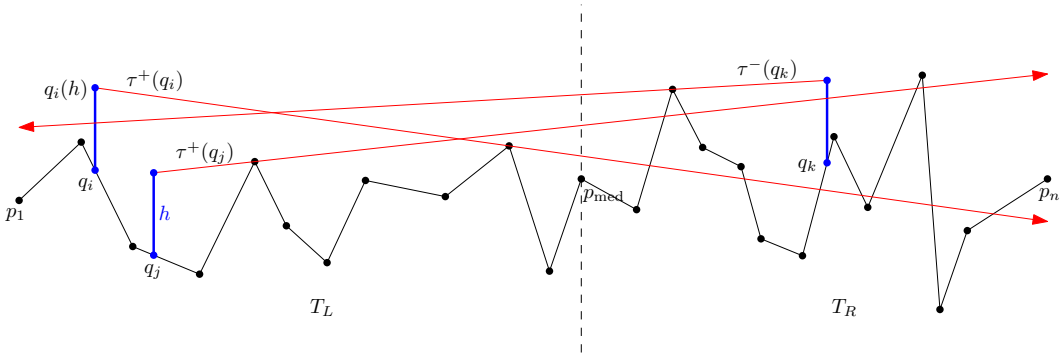

Figure 8 The criterion for visibility between a tower tip from Q_L and a tower tip from Q_R. In this example, $q_i, q_j \in Q_L$ and $q_k \in Q_R$. Moreover, since $q_k(h)$ is above the ray $\tau^+(q_i)$ and $q_i(h)$ is above the ray $\tau^-(q_k)$, q_i and q_k are mutually visible, i.e., $q_iq_k \in G(h)$. On the other hand, since $h(q_j)$ is not above $\tau^-(q_k)$, $q_jq_k \notin G(h)$.

constructions is $O(n \log n)$. Then, for each $q \in Q_L$, we express the portion of T_L to the right of q as the disjoint union of $O(\log n)$ subtrees. We compute the upper tangents from $q(h)$ to each of these hulls, and output the tangent with the largest slope. Again, with some care (including fractional cascading), this takes $O(\log n)$ time for each point of Q_L, for a total cost of $O(m_L \log n)$. A fully symmetric procedure, whose cost is $O(n \log n + m_R \log n)$, computes all the tangents $\tau^-(q')$, for $q' \in Q_R$.

To find the mutually visible pairs in $Q_L \times Q_R$, we use a two-level batched halfplane range searching algorithm, where in the first (resp., second) level the points are those of Q_R (resp., Q_L) and the halfplanes are those that lie above the lines supporting the tangents $\tau^+(q)$, for $q \in Q_L$ (resp., $\tau^-(q')$, for $q' \in Q_R$). Using standard techniques, this can be done in time $O^*(m^{4/3})$. This yields a bipartite clique decomposition of the visible edges in $Q_L \times Q_R$, and the overall collection of these graphs, over all levels of the recursion, is the desired representation of $G(h)$. As is easily checked, the overall size of the vertex sets of these graphs is $O^*(m^{4/3})$, and the overall construction takes $O^*(n + m^{4/3})$ time.

The rest of the algorithm proceeds as in Section 2. We apply BFS to $G(h)$ to determine whether s and t are connected by a path of length at most k, and, depending on the answer, we generate the next value for the binary search. We thus conclude:

▶ **Theorem 4.** *The RSP problem among m towers over a 1.5-dimensional terrain with n vertices can be solved in $O^*(n + m^{4/3})$ time.*

References

1 P. K. Agarwal. Simplex range searching and its variants: A review. In *Journey through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 1–30. Springer Verlag, Berlin-Heidelberg, 2017.

2 P. K. Agarwal, B. Aronov, E. Ezra, and J. Zahl. An efficient algorithm for generalized polynomial partitioning and its applications. *SIAM J. Comput.*, 50:760–787, 2021.

3 P. K. Agarwal, B. Aronov, M. Sharir, and S. Suri. Selecting distances in the plane. *Algorithmica*, 9:495–514, 1993.

4 P. K. Agarwal, J. Matoušek, and M. Sharir. On range searching with semialgebraic sets II. *SIAM J. Comput.*, 42:2039–2062, 2013.

5 P. K. Agarwal, M. H. Overmars, and M. Sharir. Computing maximally separated sets in the plane. *SIAM J. Comput.*, 36(3):815–834, 2006.

6 P. K. Agarwal and K. R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete Comput. Geom.*, 23(2):273–291, 2000.

7 S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics 10. Springer-Verlag, Berlin, 2nd edition, 2006.

8 H. Breu and D. G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Comput. Geom.*, 9(1-2):3–24, 1998.

9 D. Burton and P. L. Toint. On an instance of the inverse shortest paths problem. *Math. Program.*, 53:45–61, 1992.

10 S. Cabello and M. Jejčič. Shortest paths in intersection graphs of unit disks. *Comput. Geom. Theory Appls.*, 48:360–367, 2015.

11 T. M. Chan. On enumerating and selecting distances. *Int. J. Comput. Geom. Appl.*, 11(3):291–304, 2001.

12 T. M. Chan and D. Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *27th Internat. Sympos. on Algorithms and Computation*, pages 24:1–24:13, 2016.

13 T. M. Chan and D. Skrepetos. All-pairs shortest paths in geometric intersection graphs. *J. Comput. Geom.*, 10(1):27–41, 2019.

14 T. M. Chan and D. Skrepetos. Approximate shortest paths and distance oracles in weighted unit-disk graphs. *J. Comput. Geom.*, 10(2):3–20, 2019.

15 B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3):165–177, 1990.

16 T. Cui and D. S. Hochbaum. Complexity of some inverse shortest path lengths problems. *Networks*, 56(1):20–29, 2010.

17 G. D. da Fonseca, V. G. P. de Sá, and C. M. H. de Figueiredo. Shifting coresets: Obtaining linear-time approximations for unit disk graphs and other geometric intersection graphs. *Int. J. Comput. Geom. Appl.*, 27(4):255–276, 2017.

18 M. B. Dillencourt, D. M. Mount, and N. S. Netanyahu. A randomized algorithm for slope selection. *Int. J. Comput. Geom. Appl.*, 2(1):1–27, 1992.

19 A. V. Fishkin. Disk graphs: A short survey. In *First Internat. Workshop on Approximation and Online Algorithms*, volume 2909 of *Lecture Notes in Computer Science*, pages 260–264, 2003.

20 J. Gao and L. Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM J. Comput.*, 35(1):151–169, 2005.

21 M. J. Katz and M. Sharir. Efficient algorithms for optimization problems involving distances in a point set. In arXiv:2111.02052.

22 M. J. Katz and M. Sharir. An expander-based approach to geometric optimization. *SIAM J. Comput.*, 26:1384–1408, 1997.

23 J. Matoušek. Randomized optimal algorithm for slope selection. *Inf. Process. Lett.*, 39(4):183–187, 1991.

24 J. Matoušek and Z. Patáková. Multilevel polynomial partitions and simplified range searching. *Discrete Comput. Geom.*, 54:22–41, 2015.

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

26 L. Roditty and M. Segal. On bounded leg shortest paths problems. *Algorithmica*, 59(4):583–600, 2011.

27 H. Wang and J. Xue. Near-optimal algorithms for shortest paths in weighted unit-disk graphs. *Discrete Comput. Geom.*, 64(4):1141–1166, 2020.

28 H. Wang and Y. Zhao. Reverse shortest path problem for unit-disk graphs. In *17th Internat. Sympos. on Algorithms and Data Structures*, pages 655–668, 2021.

29 H. Wang and Y. Zhao. Reverse shortest path problem in weighted unit-disk graphs. In *16th Internat. Conf. on Algorithms and Computation*, pages 135–146, 2022.

30 J. Zhang and Y. Lin. Computation of the reverse shortest-path problem. *J. Glob. Optim.*, 25(3):243–261, 2003.

A Additional applications

A.1 Reverse shortest paths in proximity graphs of polylines in the plane

This is a generalization of the segment proximity problem. The input consists of a set $\mathcal{T}$ of n pairwise disjoint polylines, each of size (number of vertices) at most some constant l, two designated polylines s and t in $\mathcal{T}$, and an integer parameter $k \leq n$. The problem is to find the smallest r^* such that there exists a path between s and t of length at most k in the graph $G(r^*)$ over $\mathcal{T}$, in which there is an edge between polylines T and T' if and only if the distance between T and T' is at most r^*, where the distance between two polylines T and T' is the length of the shortest segment that connects them. See Figure 3 for an illustration.

The problem can be solved in much the same way as in the case of segments, observing that the distance between two polylines is always attained either between two vertices, one of each polyline, or between a vertex of one of them and the relative interior of an edge of the other. We can therefore write the condition that the distance between two polylines is at most r as the disjoint disjunction of several predicates, each of which has the same structure as in the case of segments. The only difference is that the number of these predicates grows quadratically in l, which does not affect the asymptotic complexity since l is a constant.

Adapting the preceding analysis, we get:

▶ **Theorem 5.** *The RSP problem in the proximity graph of n pairwise-disjoint polylines in $\mathbb{R}^2$, where each polyline is of size at most l, can be solved in $O^*(n^{4/3})$ time, where the constant of proportionality depends (quadratically) on l.*

A.2 Variations of the growing segments and the segment proximity problems

We can generalize the growing segments problem (studied in Section 3.2), and the segment proximity problem (studied in Section 2), by considering various other shapes that can be placed, or grown around each site p_i, such as ellipses. Consider first the growing problem. In the simplest version, we assume that each growing shape has only one degree of freedom of growth. For example, when growing ellipses, we assume, for instance, that the ellipse grown at a site p_i is centered at p_i, has fixed directions θ_i and $\theta_i + \frac{\pi}{2}$ of its axes, and has a fixed ratio λ_i between the lengths of its major and minor axes. We denote this ellipse as $E_i(r)$, where r, the single degree of freedom of growth, is half the length of the major axis. See Figure 5(a) for an illustration.

The problem is to find the smallest value r^* of r for which the graph $G(r^*)$ has a path between two designated points, where the edges of $G(r^*)$ are those pairs (p_i, p_j) for which $E_i(r^*) \cap E_j(r^*) \neq \emptyset$.

Here we need $t = 4$ real parameters to specify an ellipse, namely the coordinates of p_i, λ_i, and θ_i. Our machinery yields an algorithm that runs in $O^*(n^{2t/(t+1)}) = O^*(n^{8/5})$ time.

We can also generalize the proximity problem of Section 2, whose input is a set of segments, by considering other shapes, such as ellipses. That is, each ellipse E_i is given by its center p_i, the lengths of its axes $a_i > b_i$, and the direction of the major axis θ_i, and we assume that these ellipses are pairwise disjoint. Given two designated ellipses s and t in the input set and a parameter k, the goal is to find the smallest value r^* of r, such that there exists a path between s and t of length at most k in the proximity graph $G(r)$ over the ellipses E_i, whose edges are all the pairs (E_i, E_j) of ellipses, such that the distance between E_i and E_j is at most r (this latter property can be expressed as a semi-algebraic predicate of constant complexity, as is easily verified). See Figure 5(b) for an illustration.

Since we need here five real parameters to specify an ellipse, we get that the RSP problem in a proximity graph of n pairwise disjoint ellipses in the plane can be solved in $O^*(n^{5/3})$ time.

In summary, we have:

▶ **Theorem 6.**

(a) *The growing ellipses problem can be solved in $O^*(n^{8/5})$ time.*

(b) *The proximity problem for n pairwise disjoint ellipses in the plane can be solved in $O^*(n^{5/3})$ time.*

Note that this is one application where we have to apply full-blown semi-algebraic range searching machinery. We were mostly able to bypass this using multi-level data structures, each of which had to deal only with halfspace range searching. Nevertheless, other, more involved, applications of our technique will have to use this more general machinery. For example, handling other shapes, for both problems, can be handled in much the same way, using semi-algebraic range searching, whose performance depends on the number of parameters needed to specify an object. Other setups include extensions of some of the problems considered in this work to three (or higher) dimensions. For example, the predicate needed for the segment-proximity problem in $\mathbb{R}^3$ will lead to semi-algebraic ranges (namely, cylinders).

B Mixed-dimensional batched range searching

For the sake of completeness we sketch an algorithm for handling mixed-dimensional batched range searching (in a single level of a multi-level range searching structure).

In general, we have two sets A, B of n and m objects, respectively, where the objects in A have ξ degrees of freedom and those in B have η degrees of freedom, for suitable constant parameters ξ and η. Each pair $a \in A$ and $b \in B$ defines a constant-degree semi-algebraic predicate $\Pi(a, b)$. The goal is either to determine whether there exists a pair $(a, b) \in A \times B$ such that $\Pi(a, b)$ is true, or to count the number of such pairs, or to represent all of them in some compact form, or to report all of them.

Consider the first, simplest task; the other tasks are solved similarly (in the reporting version we also incur an additive cost which is linear, or nearly linear, in the output size). The solution uses a primal-dual approach. The primal space is $\mathbb{R}^\xi$, each object of A is stored as a point, and each object $b \in B$ is stored as the semi-algebraic range $Q_b = \{a \in A \mid \Pi(a, b) \text{ is true}\}$. The dual space is $\mathbb{R}^\eta$, each object of B is stored as a point, and each object $a \in A$ is stored as the semi-algebraic range $Q_a^* = \{b \in B \mid \Pi(a, b) \text{ is true}\}$. Although the problem is symmetric, we view for concreteness the objects of A as data and those of B as queries. In the primal we use a modified version of the techniques of Agarwal, Matoušek and Sharir [4] and of Matoušek and Patáková [24]. These techniques essentially construct a hierarchical polynomial partition in $\mathbb{R}^\xi$ until one reaches nodes with input size x (that we will shortly determine). The overall number of nodes is $O(n/x)$, the preprocessing time to construct the structure is $O^*(n)$, and a query with a range Q_b reaches $O^*((n/x)^{1-1/\xi})$ leaf nodes. At each leaf node we pass to the dual η-dimensional space, and apply the point-enclosure technique of Agarwal et al. [2], which processes the x dual ranges Q_a^* into a data structure of size $O^*(x^\eta)$, in time $O^*(x^\eta)$, so that a point enclosure query (with a dual point $b \in B$) takes $O(\log x)$ time.

The overall storage used by the structure, and the time to construct it, are both $O^*((n/x) \cdot x^\eta) = O^*(nx^{\eta-1})$, and a query takes $O^*((n/x)^{1-1/\xi} \log x) = O^*((n/x)^{1-1/\xi})$ time. We put $nx^{\eta-1} = \sigma$, for a suitable storage parameter σ that we will shortly determine, so $x = (\sigma/n)^{1/(\eta-1)}$. The query time then becomes

$$O^*((n/x)^{1-1/\xi}) = O^*((n^{\eta/(\eta-1)}/\sigma^{1/(\eta-1)})^{1-1/\xi}).$$

The overall cost of answering m queries, plus the preprocessing cost, is thus

$$O^*(mn^{\eta(\xi-1)/(\xi(\eta-1))}/\sigma^{(\xi-1)/(\xi(\eta-1))} + \sigma).$$

We now balance the two terms by choosing $\sigma = m^{\xi(\eta-1)/(\xi\eta-1)} n^{\eta(\xi-1)/(\xi\eta-1)}$, and conclude that the overall running time of the algorithm is

$$O^*\left(m^{\xi(\eta-1)/(\xi\eta-1)} n^{\eta(\xi-1)/(\xi\eta-1)}\right). \tag{1}$$