# Efficient Algorithms for the Bottleneck Path Problem in Geometric Graphs

**Matthew J. Katz**
The Stein Faculty of Computer and Information Science, Ben-Gurion University of the Negev, Beer Sheva, Israel

**Rachel Saban**
The Stein Faculty of Computer and Information Science, Ben-Gurion University of the Negev, Beer Sheva, Israel

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

## Abstract

We present efficient algorithms for the bottleneck path problem in two geometric settings that arise naturally in applications: directional-antenna graphs in the plane with antenna angles bounded from below by a constant, and visibility graphs whose vertices lie on or above a 1.5-dimensional terrain, both with Euclidean distances as edge weights. We provide near-linear algorithms for the corresponding decision problems, namely, determining whether the subgraph obtained by retaining all edges with weight at most some threshold $\mathbf{bn}$ contains a path from $s$ to $t$. We then use the decision procedures to obtain algorithms for the bottleneck path problem that run in $O^*(n^{8/7})$ randomized expected time, where $n$ is the input size and the $O^*(\cdot)$ notation hides subpolynomial factors.

Within the same performance bounds, we can also solve the bounded-hop version, in which we only consider $s$-$t$ paths with at most $k$ edges, for a given integer $k < n$.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Bottleneck path, BFS, Antennas, Terrain, Visibility, Bichromatic closest pair, Bounded hop, Reverse shortest paths

**Digital Object Identifier** 10.4230/LIPIcs.MFCS.2026.31

**Related Version** *Full Version*: https://arxiv.org/abs/2608.04203

**Funding** *Matthew J. Katz*: Partially supported by Israel Science Foundation Grant 495/23.
*Rachel Saban*: Partially supported by the Lynne and William Frankel Center for Computer Science and by Israel Science Foundation Grant 495/23.
*Micha Sharir*: Partially supported by Israel Science Foundation Grant 495/23.

## 1 Introduction

Let $G = (V, E)$ be a weighted (and possibly directed) graph, and let $w$ be a positive weight function on the edge set $E$. The *bottleneck* of a path $\pi$ in $G$, denoted $\mathbf{bn}(\pi)$, is the weight of the heaviest edge of $\pi$. In the *bottleneck path* problem, we are given two vertices $s, t \in V$, and the goal is to find a path from $s$ to $t$ with minimum bottleneck (assuming there exists a path from $s$ to $t$ in $G$). We denote the bottleneck of such an optimal path by $\mathbf{bn}(s, t)$, that is, $\mathbf{bn}(s, t) = \min\{\mathbf{bn}(\pi) \mid \pi \text{ a path in G from s to t}\}$.

Consider the associated *decision problem*, which receives as input a threshold $\mathbf{bn}$, and aims to determine whether $G$ contains a path $\pi$ from $s$ to $t$ with $\mathbf{bn}(\pi) \leq \mathbf{bn}$, that is, whether $G_{\leq \mathbf{bn}} = \{(u, v) \in G \mid w(u, v) \leq \mathbf{bn}\}$ has a path from $s$ to $t$. In the problems that we study, we are able to solve that problem in near-linear time, but we do not know how to achieve such a running time for the bottleneck path problem itself; see below how we tackle such situations.

We also consider the *bounded-hop* version of the problem, in which we are also given an integer parameter $1 \leq k < n$, and wish to determine, in the decision version, whether $G_{\leq \mathbf{bn}}$ contains a path from $s$ to $t$ with at most $k$ edges. In the optimization version, i.e., the bottleneck path problem, we seek the minimum value of $\mathbf{bn}$ for which $G_{\leq \mathbf{bn}}$ contains a path from $s$ to $t$ with at most $k$ edges. This is also known as the *reverse shortest path* (RSP) problem for $G$.

For example, consider the case where $G$ is a Euclidean graph (some suitably defined subgraph of the complete graph) on a set $P$ of $n$ points in the plane. In this case we can sometimes solve the decision problem in near-linear time (it is not always an easy task, and depends on how $G$ is specified), but even in such cases, when the constraints defining $G$ are nontrivial, we do not know how to solve the bottleneck path problem (the optimization version of the problem) in near-linear time, as it is generally based on running BFS on the threshold graph, and we do not know how to simulate the execution of BFS in parallel, in the standard style of parametric search.[1] A simple solution to that latter problem is to use a distance selection algorithm (such as in [2]) to perform a binary search through all $\binom{n}{2}$ inter-point distances (the optimal $\mathbf{bn}$ is one of these distances), using the decision procedure to guide the search. This yields an $O^*(n^{4/3})$-time algorithm for the bottleneck path problem (for points in the plane), where the $O^*(\cdot)$ notation hides subpolynomial factors.

However, in such cases we can improve the running time, when the decision procedure has a near-linear solution, by applying the shrink-and-bifurcate machinery of Ben Avraham et al. [7] with the recent improvements of Chan and Huang [11] (see also [14]), to solve the bottleneck path problem in $O^*(n^{8/7})$ randomized expected time. See later in the paper for details.

We present two applications of the latter approach, where the major challenge is to find appropriate near-linear implementations of the decision procedure. Typically, as already mentioned, this procedure runs BFS on $G_{\leq \mathbf{bn}}$. In typical setups, as in our applications, we are given a geometric scene consisting of a set $P$ of points in the plane, a range[2] $r = \mathbf{bn} > 0$, and some additional constraints, which indicate which pairs of points of $P$ can "see" each other (i.e., define edges of $G_{\leq \mathbf{bn}}$). That is, the scene induces a *visibility graph* on $P$, in which there is an edge between two points $p, q \in P$ if and only if they see each other, in the sense that the segment $pq$ satisfies the constraints imposed on the problem, and are within distance at most $r$. For example, in one of the applications, each point $p \in P$ represents a directional antenna, i.e., a wedge $W_p$ with apex $p$, and there is an edge between $p$ and $q$ if and only if $q \in W_p$ (in the directed case) or $q \in W_p$ and $p \in W_q$ (in the undirected case), and the distance between $p$ and $q$ is at most $r$.

The latter approach is also suitable for the aforementioned *bounded-hop* variant of the bottleneck path problem in $G_{\leq \mathbf{bn}}$, where, in addition, we are given a parameter $1 \leq k < n$ and the goal is to find a path from $s$ to $t$ consisting of at most $k$ edges, with minimum bottleneck.

Another variant of this scheme involves bottleneck paths in the visibility graph of a set $P$ of $m$ points lying on or above a 1.5-dimensional terrain (an $x$-monotone polygonal path) with $n$ vertices in the plane. A significantly simpler version of this problem, without any bounded-range constraints, was recently studied in [16]. Here we are given a range $r > 0$, and say that two points $p, q \in P$ are mutually visible (over $T$) if the segment $pq$ lies fully above $T$, and $|pq| \leq r$. Here too, the bottleneck version receives two designated points $s, t \in P$, and

[1] Running BFS in small parallel depth is a general notorious open problem.
[2] In what follows, we mostly use $r$ instead of $\mathbf{bn}$.

seeks the minimum value of $r$ for which the resulting visibility graph contains a path from $s$ to $t$ (or a $k$-bounded-hop path from $s$ to $t$ in the bounded-hop version). In the decision version $r$ is fixed, and the goal is to determine whether the graph, with that $r$, contains a path from $s$ to $t$ (or a path with at most $k$ edges).

**Related work.** Breadth-First Search (BFS) is recognized as one of the fundamental graph algorithms. In general, its running time is $O(|V| + |E|)$, which is inefficient when $|E|$ is large. Its importance has led authors to seek families of graphs in which BFS can be implemented more efficiently, i.e., in subquadratic time that only depends on the number of vertices. One such example is the family of disk graphs. The vertices of a disk graph represent disks in the plane and there exists an edge between two vertices if and only if the corresponding disks intersect. Notice that the number of edges in a disk graph can be quadratic in $n$, the number of disks. Nevertheless, for unit-disk graphs, Cabello and Jejčič [10] presented an $O(n \log n)$ implementation of BFS, and subsequently Chan and Skrepetos [12] presented an alternative $O(n)$ implementation (after pre-sorting the points by their $x$- and $y$-coordinates). For arbitrary disks, Kaplan et al. [14] and Liu [18] described an $O(n \log^4 n)$ implementation of BFS, which was recently improved to $O(n \log^2 n)$ by Klost [17] and subsequently to $O(n \log n)$ by Cabello and de Berg [13].

The proximity graph of a set $S$ of $n$ segments in the plane, for a real parameter $r \geq 0$, is $G_r(S) \coloneqq (S, E)$, where $E = \{(e_1, e_2) \mid \text{dist}(e_1, e_2) \leq r\}$ and $\text{dist}(e_1, e_2)$ is the Euclidean distance between $e_1$ and $e_2$ (which is 0 if they intersect). Agarwal et al. [4] devised an $O^*(n^{4/3})$ implementation of BFS in $G_r(S)$. For the special case where the segments in $S$ are pairwise disjoint, Agarwal et al. [3] have later provided an $O(n \log^2 n)$ implementation.

The bottleneck path problem (also known as the minimax path problem) and its complementary problem, i.e., the widest path problem (or the maximum capacity problem), are well-known problems in graph theory. Abu-Affash et al. [1] studied the problem of placing at most $k$ Steiner points to minimize the bottleneck of a path between two designated points in the plane, for a given integer parameter $k \geq 0$, and developed an $O(n \log^2 n)$-time algorithm for the problem.

The reverse shortest path (RSP) problem for unit disks (mentioned above) was studied by Wang and Zhao [20], who proposed an $O^*(n^{5/4})$-time solution. An improved solution with running time $O^*(n^{6/5})$, using the shrink-and-bifurcate technique, was subsequently presented by Kaplan et al. [14]. A recent improvement of the shrink-and-bifurcate technique, due to Chan and Huang [11], allows one to improve the running time to $O^*(n^{9/8})$. The RSP problem was also studied for other objects, including wedges of some fixed angle, which are viewed as directional antennas. This latter version, which is especially relevant to our current work, was solved in $O^*(n^{4/3})$ time by Agarwal et al. [4].

Finally, wireless networks, and in particular those utilizing directional antennas, as defined above, have received significant attention in recent years; see, e.g., the series of papers dealing with the bounded-angle spanning tree problem [5, 6, 8, 9].

**Our results.** Our main contribution is in providing near-linear implementations of BFS in two types of visibility graphs arising in realistic scenarios. It is interesting and somewhat unexpected that such implementations exist, since at first sight it seems that employing simplex range searching, which would immediately increase the running time to at least $O^*(n^{4/3})$, is inevitable, when the constraints defining the graph involve halfplane or disk range searching, as in our applications, or in more general setups.

Let $P$ be a set of $n$ points in the plane, where each point $p \in P$ has a wedge $W_p$ with apex at $p$ associated with it. Let $\alpha > 0$ be such that each wedge is of angle at least $\alpha$. (This is a key assumption, without which a running time near $n^{4/3}$ seems inevitable.) Let $r > 0$ be a real parameter. The wedges, viewed as directional antennas of range $r$, naturally induce two types of graphs. The first is the directed visibility graph $\overrightarrow{VG}(P, r)$ of $P$, where there is a directed edge from $p \in P$ to $q \in P$ if and only if $q \in W_p$ and $|pq| \leq r$. The second type is the undirected visibility graph $VG(P, r)$, where there is an edge between $p$ and $q$ if and only if $p \in W_q$, $q \in W_p$, and $|pq| \leq r$. In Section 2 we devise near-linear implementations of BFS in these graphs. We then apply the shrink-and-bifurcate technique to solve the (general or bounded-hop) bottleneck path problem in these graphs in $O^*(n^{8/7})$ randomized expected time, using BFS as the decision procedure. The $O^*(n^{8/7})$ bound improves the $O^*(n^{4/3})$ bound of Agarwal et al. [4], mentioned earlier, which holds in a slightly more general setting.

We also study bounded-range visibility graphs of points on or above a 1.5-dimensional terrain $T$ (i.e., an $x$-monotone polygonal curve). That is, there is an edge between two points $p$ and $q$ if and only if the line segment $pq$ lies above $T$ and $|pq| \leq r$. Recently, Katz et al. [16] presented a near-linear implementation of BFS in such graphs without the bounded-range constraint (i.e., assuming $r = \infty$). In Section 3, we present a near-linear implementation for any finite $r$, which is a significantly more difficult problem. Here too, by combining this procedure with the shrink-and-bifurcate technique, we solve the corresponding bottleneck path problem in $O^*(n^{8/7})$ randomized expected time.

# 2 The case of antennas

## 2.1 BFS in directed visibility graphs induced by directional antennas

Let $P$ be a set of $n$ points in the plane, each representing a *directional antenna* of angle at least $\alpha$, for some $\alpha > 0$. Specifically, each point $p \in P$ has a wedge $W_p$ associated with it, whose apex is $p$, with an opening angle at least $\alpha$. Let $r$ be a real parameter and let $VG(P, r)$ be the directed visibility graph of $P$, where there is a directed edge $(p, q)$ from $p \in P$ to $q \in P$ if and only if $q \in W_p$ and the distance between $p$ and $q$ is at most $r$ (in sensor-network parlance, $q$ is within the transmission range of $p$); see Figure 1. In this section we present a near-linear algorithm for performing BFS in the graph $VG(P, r)$, from a given start point $s \in P$. This is nearly the best algorithm one can hope for, recalling that the graph may have quadratically many edges.

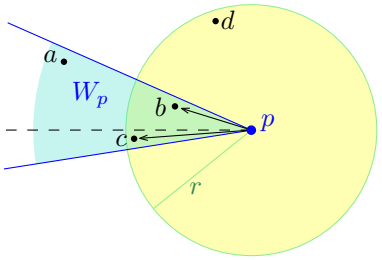


**Figure 1** The points $b$ and $c$ lie in $W_p$ and are at distance at most $r$ from $p$. The point $a$ is out of range, and $d$ is not in $W_p$.

The general scheme goes as follows. As usual, we construct the BFS layer by layer; the $i$-th layer is denoted by $L_i$, starting with $L_0 = \{s\}$. Suppose we have already constructed $L_i$ and now want to construct $L_{i+1}$. We maintain the set of points of $P$ that the BFS has not reached yet; denote by $P_i$ this set at the end of the $i$-th iteration.

We construct and maintain the following dynamic data structure on $P_i$. We query it with the points of $L_i$, where the goal of the queries is to find and report all the points $p \in P_i$ that are visible from at least one point of $L_i$ (in the sense that there exists $q \in L_i$ such that $p \in W_q$ and $|pq| \leq r$), and delete them from the structure as soon as they are discovered, to ensure that no point is reported more than once. By definition, these points comprise $L_{i+1}$. Their deletion from $P_i$ yields $P_{i+1}$.

Dropping the index $i$ for convenience, the setup is thus as follows. We have a set $P$ of points in the plane, which we want to maintain under deletions, so as to be able to answer efficiently queries, each of which specifies a query point $q$ (representing an antenna), and seeks to report all the points of $P$ that are visible from $q$ (i.e., lie in $W_q$), and at distance at most $r$ from it. The problem we face is that the condition for $qp$ to be an edge is a conjunction of three constraints: $p$ has to (i) lie counterclockwise to the clockwise ray bounding $W_q$, (ii) lie clockwise to the counterclockwise ray bounding $W_q$, and (iii) be at distance $\leq r$ from $q$. A naïve, albeit inefficient, implementation would use a three-level (dynamic) reporting data structure, which would result in a significantly superlinear algorithm. To avoid this inefficiency, we adopt the following approach. The idea is to partition the plane into axis-aligned regions, which induce a partition of $P$ into subsets. We then show that for each such subset it is sufficient to check only one of the above three constraints. That is, in some of the subsets it suffices to report all points at distance at most $r$ from $q$ (using a dynamic Voronoi diagram), in other subsets it suffices to report all points that are counterclockwise to the clockwise ray of $W_q$, and in other subsets it suffices to report all points that are clockwise to the counterclockwise ray of $W_q$ (using a dynamic halfplane range reporting data structure in each of the two latter cases). In each of these three cases, the other two constraints are automatically satisfied.

For this, we prepare the following (dynamic) multi-level data structure for the subsets of $P$ in each of the regions. We first construct a two-level dynamic orthogonal range tree, where the primary tree $T$ is constructed on the $y$-order of the points of $P$, and at each node $v$ of $T$, we construct a secondary tree $T_v$ on the points stored at the subset of $T$ rooted at $v$, sorted by their $x$-coordinates. Finally, at each node $\eta$ of each $T_v$, we create two dynamic third-level structures on the subset $P_{v,\eta}$ of the points stored at the subtree rooted at $\eta$. One structure is a dynamic Voronoi diagram of $P_{v,\eta}$, and the other is a dynamic halfplane range reporting structure for $P_{v,\eta}$. The Voronoi diagram is constructed and maintained using the technique of Kaplan et al. [15], improved by Liu [18]. The halfplane range reporting structure is constructed and maintained using the dynamic algorithm of Overmars and van Leeuwen [19]. In the latter structure, a deletion costs $O(\log^2 n)$ time, and a query takes $O(\log n)$ time, which are dominated by the $O(\log^4 n)$ time for searching and updating the Voronoi diagram. We note that these running time bounds for deletions are amortized.

Finally, we maintain $\lceil \frac{\pi}{\alpha} \rceil$ copies of the data structure, denoted $T_0, T_1, T_2, \ldots$, where $T_i$ stores a copy of $P$ obtained by rotating the original scene clockwise about the origin by the angle $i\alpha$, so that lines of (original) orientation $i\alpha$ become horizontal. Given a query point $q$ associated with a wedge $W_q$, we perform the query in the copy $T_i$, where $i$ is the smallest index for which $W_q$ is intersected by the line through $q$ at orientation $i\alpha$ (actually, any index $i$ with this property will do). To perform the query in $T_i$, we rotate the wedge $W_q$ clockwise by $i\alpha$ degrees, so that $W_q$ contains either the rightward or the leftward horizontal ray emanating from $q$. See Figure 2.

For convenience, we refer to the chosen data substructure as $T$. Given a query point $q$ with associated wedge $W_q$ that contains a horizontal ray emanating from $q$, we divide $W_q$ by horizontal and vertical lines through $q$, into at most five sub-wedges, each of angle at most

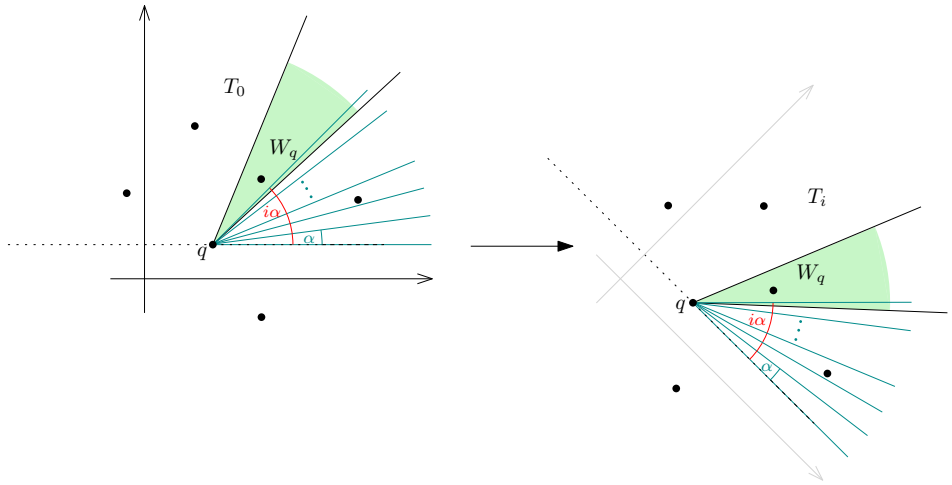


**Figure 2** Left: The original scene and a query wedge $W_q$. Right: The query will be performed in the $i$'th copy $T_i$, after rotating $W_q$ clockwise by $i\alpha$ degrees.

$\pi/2$ and with at least one axis-parallel ray; see Figure 3. We perform the query on each sub-wedge separately. Without loss of generality, we present here the query process for a sub-wedge whose lower ray is a horizontal right ray.

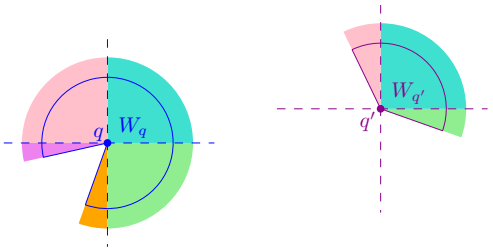


**Figure 3** Horizontal and vertical lines through a point divide its wedge into at most five sub-wedges, each of angle at most $\pi/2$ and with at least one axis-parallel ray.

For convenience, we refer to this sub-wedge also as $W_q$. Let $m^+(q)$ be the point at distance $r$ from $q$ along the upper ray of $W_q$; see Figure 4. We first search in the first level of $T$ with (the $y$-coordinates of) $q$ and $m^+(q)$. This yields all the points of $P$ lying in the slab $\sigma$ bounded by the horizontal lines through $q$ and $m^+(q)$, as the disjoint union of $O(\log n)$ canonical sets, each being the subset stored at some subtree of $T$. At each such subtree, rooted at some node $v$, we search in the secondary tree $T_v$ with the $x$-coordinates of $q$ and $m^+(q)$, thereby splitting $\sigma$ into three rectangular regions, a semi-unbounded region $Q_L$ lying to the left of $q$, a bounded rectangle $Q_M$ lying between $q$ and $m^+(q)$, and another semi-unbounded region $Q_R$ lying to the right of $m^+(q)$. Notice that in case $W_q$ is of angle $\pi/2$, $Q_M$ is empty. Put $P_L = P \cap Q_L$, $P_M = P \cap Q_M$, and $P_R = P \cap Q_R$. Clearly, no point of $P_L$ can lie in $W_q$.

For $P_R$, all its points lie in $W_q$, so it suffices to report those points at distance $\leq r$ from $q$. To do so, we query with $q$ at the corresponding dynamic Voronoi diagrams of the $O(\log^2 n)$ canonical subsets $P_{v,\eta}$ that comprise $P_R$. Each output point $p$ at distance $\leq r$ from $q$ is added to $L_{i+1}$, and is promptly deleted from all the $O(\log^2 n)$ structures (Voronoi diagrams and dynamic halfplane range reporting structures) that it participates in. We stop querying with $q$ when no neighbor at that distance is found.

For $P_M$, all its points are at distance $\leq r$ from $q$, so it suffices to report those points that lie in $W_q$. By construction, this is equivalent to reporting all the points of $P_M$ that lie below the upper ray of $W_q$. We follow the same approach as before, but now we query at the halfplane range reporting structures corresponding to the relevant canonical subsets.

Note that no point above the horizontal line through $m^+(q)$ is visible from $q$ – it either is too far from $q$ or lies above $W_q$. See Figure 4 for an illustration.

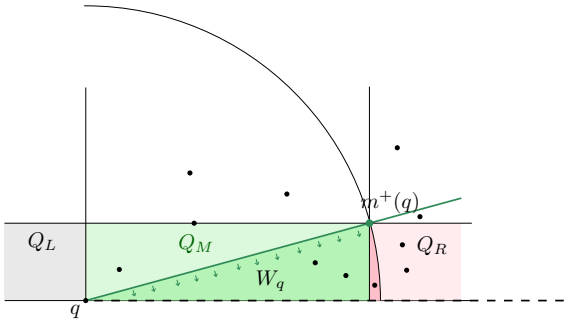


**Figure 4** The rectangular regions arising when querying with a point $q$, for the points above $q$. $m^+(q)$ is the point at distance $r$ from $q$ along the upper ray of $W_q$. Points in $P_R = P \cap Q_R$ lie in $W_q$, and points in $P_M = P \cap Q_M$ are at distance at most $r$ from $q$. A symmetric partition holds for the points below $q$.

Symmetric procedures are applied for each of the other cases – sub-wedges below the horizontal line through $q$, sub-wedges for which the axis-parallel bounding ray is vertical, and so on.

As there are $O(\log^2 n)$ canonical subsets that the query processes, the overall cost of a query (i.e., the part of the query that finds a neighbor of $q$ or determines that no such neighbor exists), followed by deletion when a neighbor is found, is $O(\log^6 n)$ (which is amortized for deletions).

The correctness of the query procedure is straightforward, and follows from the arguments given above.

We now run BFS on $VG(P, r)$, using the above data structure, in the manner described earlier. Each point of $P$ acts as a query only at the layer it belongs to. The cost of querying with a point $q$ is $O((1 + k_q) \log^6 n)$, where $k_q$ is the number of (yet unvisited) neighbors of $q$ that the algorithm detects. Since no point is reported more than once (since it is deleted from all its containing structures once it is discovered), we have $\sum_q k_q \leq n$, so the overall cost of the algorithm is $O(n \log^6 n)$. That is, we have shown:

▶ **Theorem 1.** *BFS on $VG(P, r)$ can be performed in $O(n \log^6 n)$ time.*

## 2.2 The corresponding bottleneck path problem

Recall that the bottleneck path problem is to find the minimum value $r^*$ for which there exists a path from $s$ to $t$ in $VG(P, r^*)$. This optimization problem can be solved using the shrink-and-bifurcate technique of [7, 11, 14], as mentioned in the introduction, using the BFS algorithm of Theorem 1 as the decision procedure.

**A brief review of the shrink-and bifurcate technique.** In more details, this technique is applied when we have an efficient decision procedure, for which we do not have an efficient parallel implementation to facilitate standard parametric search. What we do is first run

an interval-shrinking phase, which produces an interval $I$ that contains a small number of critical values, including the optimal value $r^*$. We then simulate the decision procedure sequentially, where most of the comparisons (those whose critical values lie outside $I$) can be resolved. For those comparisons with critical values in $I$, we bifurcate, following both outcomes of the comparison. We stop the bifurcation when we exceed a certain threshold number of them, or when the simulation has executed at least some threshold number of steps, and then resolve all of them using the standard binary search employed by parametric search. Then a new bifurcation stage begins, at the leaf found to contain $r^*$, and we proceed this way until we simulate the entire decision procedure.

A careful choice of parameters makes this procedure efficient. For example, when the critical values are inter-point distances in the plane, this shrink-and-bifurcate procedure can be implemented to run in $O^*(n^{8/7})$ randomized expected time [11].

This sketchy review only provides some highlights of the technique. See [7] and [16] for full details, and see [11] for a recent improved implementation of the shrinking phase.

**Applying the shrink-and-bifurcate technique.** In our context, the critical values of $r$, namely values at which the outcome to some comparison made by the BFS changes, are of two kinds: (i) pairwise distances between the points of $P$, and (ii) values at which some point $m^+(q)$ (or its symmetric counterpart along the lower ray of $W_q$) becomes co-vertical or co-horizontal with some point of $P$. We handle each kind of critical values separately. We begin the procedure by sorting all the points of $P$ into the two-dimensional orthogonal range tree $T$. We then locate the $y$-coordinates of the points $m^+(q)$ in the primary tree, to obtain the canonical subsets that represent the points below and above each $m^+(q)$.

The simulation of this step, with the unknown $r^*$, can be performed using standard parametric search, since we can search with the points $m^+(q)$ in parallel, and each search only takes $O(\log n)$ steps.

Similarly, for each canonical set $T_v$ that we obtain for some $m^+(q)$, we search with the $x$-coordinate of $m^+(q)$ in $T_v$, to obtain a representation of the points of $T_v$ to the left and to the right of $m^+(q)$. This too can be done in parallel, over all points $q$, all nodes $v$, and the $O(\log n)$ steps of each binary search.

A symmetric procedure is applied to the corresponding points on the lower rays.

Hence, when this part of the simulation ends, all critical values of type (ii) are out of the interval that contains $r^*$, as obtained by the standard parametric search technique, and each (future) comparison that involves them can be resolved, independently of $r^*$.

For critical values of type (i), we use the shrink-and-bifurcate technique of [7,14] reviewed above. Using the implementation of Chan and Huang [11] for the shrinking part of the procedure yields a procedure that runs in $O^*(n^{8/7})$ randomized expected time.

Note that the same technique can also solve, with the same performance bounds, the aforementioned bounded-hop or RSP (reverse shortest path) version of the problem, in which we are given an additional integer parameter $k$, and seek the minimum $r^*$ for which $VG(P, r^*)$ contains a path from $s$ to $t$ of length at most $k$. That is, we have:

▶ **Theorem 2.** *The bottleneck path problem, as well as its bounded-hop (RSP) version, for directed visibility graphs induced by $n$ directional antennas in the plane, each of opening angle at least $\alpha$, for some constant parameter $\alpha > 0$, can be solved in $O^*\left(\frac{1}{\alpha}n^{8/7}\right)$ randomized expected time.*

## 2.3 BFS in undirected visibility graphs induced by directional antennas

Let $r, \alpha$ be positive real parameters and let $P$ be a set of $n$ points in the plane, each representing a directional antenna, with an associated wedge $W_p$ of angle $\geq \alpha$. We consider here the setup, in which two antennas, with apices $p$, $q$, can see (or rather communicate with) each other if and only if $q \in W_p$, $p \in W_q$, and $|pq| \leq r$. The corresponding undirected graph $VG(P, r)$ consists of all mutually visible pairs $(p, q)$. Given two antennas $s, t \in P$, the goal is to determine whether $t$ is reachable from $s$ in $VG(P, r)$, or, in the bounded-hop version, whether $t$ is reachable from $s$ along a path with at most $k$ edges, for an additional parameter $k$.

To solve the reachability problem, we use a similar scheme to that in the directed version, modifying only the query process. Following the procedure described in Section 2.1, without loss of generality, we present the query process for a query point $q$ and its sub-wedge of angle at most $\pi/2$ whose lower ray is a horizontal right ray. That is, we detect only the visible points lying above and to the right of $q$. As in the directed version, a symmetric process is applied to each of the other sub-wedges of $W_q$.

Notice that for each antenna $p$ lying above and to the right of $q$, the point $q$ lies in the lower-left quadrant of $p$. Therefore, we handle antennas in different ways, depending on how their wedge intersects this lower-left quadrant. See Figure 5, where the lower-left quadrant of $p$ is denoted by $L_p$.

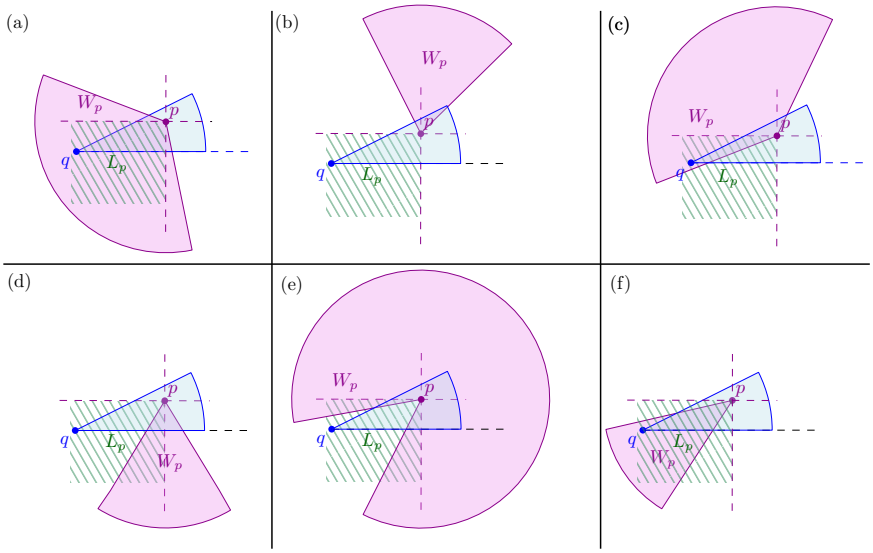


**Figure 5** The query in the undirected case. (a) $L_p \subseteq W_p$, i.e., $q \in W_p$. (b) $L_p \cap W_p = \emptyset$, in which case $p$ and $q$ do not see each other. (c) $L_p \cap W_p$ is a sub-wedge that contains the left-horizontal ray from $p$. (d) $L_p \cap W_p$ is a sub-wedge that contains the lower-vertical ray from $p$. (e) $L_p \cap W_p$ is the union of two disjoint sub-wedges, one of type (c) and one of type (d). (f) $W_p \subset L_p$, so that it does not contain any axis-parallel ray from $p$.

- For points $p$ with $L_p \subseteq W_p$ – guaranteeing that $q \in W_p$ (Figure 5(a)) – it suffices to detect every $p$ lying in $W_q$ at distance at most $r$ from $q$. This coincides with the visibility definition in the directed version, and we simply follow the corresponding query process described there.

- If $L_p \cap W_p = \emptyset$ (Figure 5(b)), we can safely conclude that $q \notin W_p$, and therefore $p$ and $q$ are not visible to each other.
- The case where $W_p$ contains only the horizontal ray bounding $L_p$ but not the vertical one is referred to as **case (c)** in Figure 5, and is studied in Section 2.3.1.
- The case where $W_p$ contains only the vertical ray bounding $L_p$, but not the horizontal one, referred to as **case (d)** in Figure 5, is symmetric to the former case (c).
- The case where $W_p \cap L_p$ is the union of two disjoint sub-wedges (Figure 5($e$)), one of type (c) and the other of type (d), is handled by applying the corresponding procedure to each sub-wedge separately.
- Finally, consider the points $p$ with $W_p \subset L_p$, where $W_p$ contains no axis-aligned rays, see Figure 5($f$). In Section 2.3.2 we present a further multi-level partition of these points into $\frac{\pi}{2\alpha}$ subsets, and each subset is split into points that can be handled as in case (c), and points that can be handled as in case (d).

### 2.3.1 Handling case (c)

In this case $p$ lies above the lower ray of $W_q$, and $q$ lies below the upper ray of $W_p$, so it suffices to test the position of $p$ with respect to the upper ray of $W_q$, and the position of $q$ with respect to the lower ray of $W_p$. Let $\sigma$ be the counterclockwise angle between the rightward-directed horizontal ray emanating from $q$ and $qp$ (the same as between the leftward-directed horizontal ray emanating from $p$ and $pq$), see Figure 6.

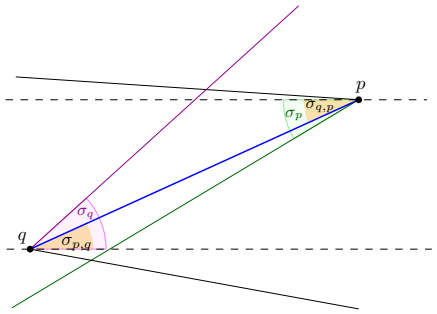


**Figure 6** For $p$ lying above $q$, $p$ and $q$ lie in each other's wedge if and only if $p$ is below the upper ray of $W_q$ and $q$ is above the lower ray of $W_p$. Moreover, if $\sigma_p \leq \sigma_q$, as in the figure, then for $p$ and $q$ to lie in each other's wedge it suffices that $q$ be above the lower ray of $W_p$. A symmetric single condition holds when $\sigma_p \geq \sigma_q$.

Denote by $\sigma_p$ (resp., $\sigma_q$) the angle of the sub-wedge at $p$ (resp., at $q$) between its lower ray (resp., upper ray) and the $x$-axis. Observe that $p$ lies below the upper ray of $W_q$ if and only if $\sigma \leq \sigma_q$, and $q$ lies above the lower ray of $W_p$ if and only if $\sigma \leq \sigma_p$. It therefore suffices to test whether $\sigma \leq \min\{\sigma_q, \sigma_p\}$.

We therefore distinguish between two main cases, and handle each of them separately. First, consider those $p$ with $\sigma_q \leq \sigma_p$. In this case if $p$ lies below the upper ray of $W_q$, then $q$ certainly lies above the lower ray of $W_p$, hence we report all the points that lie below the upper ray of $W_q$, and are at distance at most $r$ of $q$. This is done exactly as in Section 2.1; see Figure 4.

Next, consider those $p$ with $\sigma_p \leq \sigma_q$ (this is the case depicted in Figure 6). Here we need to report all the points $p$ at distance at most $r$ from $q$, such that $q$ lies above the lower ray of $W_p$ (and then $p$ certainly lies below the upper ray of $W_q$). This case is more involved and is handled as follows. Consider the partition of the plane into four quadrants obtained by

drawing a horizontal and a vertical line through $q$. We denote the quadrants starting from the northeast one and advancing clockwise by $A$, $E$, $F$, and $G$, respectively; see Figure 7. Recall that we are only interested in points $p$ that lie above and to the right of $q$, i.e., in $A$.

Let $m^-(p)$ be the point that lies at distance $r$ from $p$ along the lower ray of $W_p$, and distinguish between several cases, depending on which of the four quadrants $A, E, F, G$ defined by $q$ contains $m^-(p)$.

If $m^-(p)$ lies in the southeast quadrant $E$ (see points $p_5$ and $p_6$ in Figure 7), then clearly $q$ lies above the lower ray of $W_p$, and it suffices to test whether $|pq| \leq r$, by searching in the suitable Voronoi diagrams.

If $m^-(p)$ lies in the northwest quadrant $G$ (see $p_2$ in Figure 7) then clearly $q$ lies below the lower ray of $W_p$, so this case can be ignored.

Consider the case where $m^-(p)$ lies in the southwest quadrant $F$ (see $p_3$ and $p_4$ in Figure 7). Then both segments $m^-(p)q$ and $qp$ have positive slopes, implying that the angle $\angle m^-(p)qp$ between them is obtuse. Therefore, $|pq| < |pm^-(p)| = r$, so it suffices to test whether $q$ lies above the lower ray of $W_p$. (This step is implemented in the dual plane, as a halfplane range reporting query with the line dual to $q$ amid the points dual to the lines containing the lower rays of the appropriate set of yet undiscovered antennas with a leftward-downward directed lower ray.)

Finally, consider the case where $m^-(p)$ lies in the northeast quadrant $A$ (see $p_1$ in Figure 7). Here both segments $qm^-(p)$ and $m^-(p)p$ have positive slopes, implying that the angle $\angle qm^-(p)p$ between them is obtuse. Therefore, $|pq| > |pm^-(p)| = r$, so this case can be ignored too.

To recap, knowing the quadrant containing $m^-(p)$ allows us either to discard $p$ altogether as a possible neighbor of $q$, or to determine which of the two reporting tasks (halfplane range reporting in the dual plane or searching in Voronoi diagrams) suffices to determine whether $p$ is a neighbor of $q$.

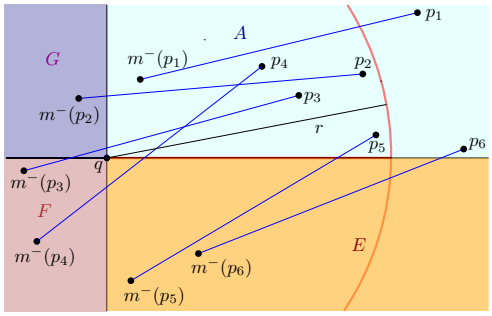


**Figure 7** Partition of the plane into regions. Points $p \in A$ and the corresponding segments $pm^-(p)$.

The actual data structure is constructed, and searched in, as follows. For concreteness, and without loss of generality, we continue to focus on the task of reporting the points $p$ that lie above and to the right of the query point $q$, and satisfy $p \in W_q$, $q \in W_p$, and $|pq| \leq r$. The tasks of reporting the points of $P$ in the other three cases are treated in a fully symmetric manner. We construct an orthogonal range tree $T_\sigma$ on the angles $\sigma_p^-$, for $p$ in the set of antennas being maintained, where $\sigma_p^-$ (formerly referred to as $\sigma_p$) is the angle between the left horizontal direction and the lower ray of $W_p$. We search in $T_\sigma$ with the angle $\sigma_q^+$, the angle between the right horizontal direction and the upper ray of $W_q$, to obtain a collection

of $O(\log n)$ canonical subsets, in each of which we know whether the angles $\sigma_p^-$ of the points $p$ in the subset are all smaller or all larger than $\sigma_q^+$. Consequently, for each of the canonical subsets of $T_\sigma$, we construct two data structures, one for each of the two cases $\sigma_p^- \leq \sigma_q^+$ and $\sigma_p^- \geq \sigma_q^+$, for all $p$ in the subset. In order to deal with the latter case, we construct and use, at each canonical set stored at some node of $T_\sigma$, the data structure of Section 2.1. In order to deal with the former case, we construct the following multilevel data structure. The first four levels of the structure constitute a four-dimensional orthogonal range tree, where the first two levels are constructed on the $y$-coordinates and $x$-coordinates of the points $p$, and the last two levels are constructed on the $y$-coordinates and $x$-coordinates of the points $m^-(p)$. Again, this is done for each canonical set stored at some node of the top level $T_\sigma$. By searching with $q$ in these four-level range trees, we determine the points $p$ above and to right of $q$, whose associated points $m^-(p)$ lie in each of the two lower quadrants determined by $q$. (By the preceding analysis we can ignore the two upper quadrants.) For the canonical subsets of points $p$ whose associated points $m^-(p)$ lie in the southeast quadrant $E$ determined by $q$, we maintain a dynamic Voronoi diagram and use it to detect and report all the points at distance at most $r$ from $q$. For those canonical subsets of points $p$ whose associated points $m^-(p)$ lie in the southwest quadrant $F$, we maintain a dynamic halfplane range reporting structure in the dual plane, to detect and report all the points $p$ for which $q$ lies above the lower ray of their wedge $W_p$. See Figure 8 for an illustration of the structure.

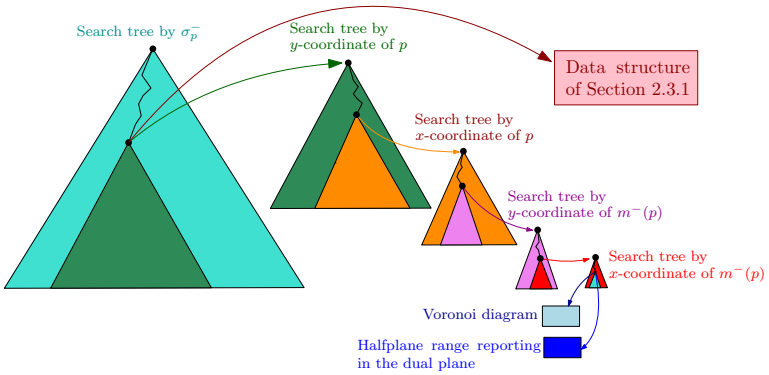


**Figure 8** A schematic illustration of the data structure used in the undirected visibility graph for case (c).

### 2.3.2 Handling case (f): $W_p \subset L_p$

Here we are given a set of points $p$ with $W_p \subset L_p$, where $W_p$ does not contain the horizontal and vertical rays of $L_p$. We partition the set into subsets of points where either all have their upper ray above $q$ (handled as in case $(c)$) or all have their lower ray below $q$ (handled as in case $(d)$). We achieve this by first dividing the set into $\ell = \lceil \frac{\pi}{2\alpha} \rceil$ subsets, where the $i$-th subset, $i = 1, \ldots, \ell$, consists of all points $p$ for which $i$ is the minimal index such that $W_p$ contains a ray in direction $d_i = \pi + i\alpha$ (recall that all wedges have angle at least $\alpha$ and each has an upper ray in a direction greater than $\pi$).

We store the $i$-th subset in a balanced binary search tree, ordered by the projections of its points onto $l_i$, the line perpendicular to $d_i$. Notice that if the projection of $p$ onto $l_i$ is higher than that of $q$, we can safely conclude that $q$ lies below the upper ray of $W_p$, and thus all such points can be handled as in case $(c)$. Symmetrically, if the projection of $p$ onto $l_i$ is lower than that of $q$, then $q$ lies above the lower ray of $W_p$, and all such points can be handled as in case $(d)$ (see Figure 9).

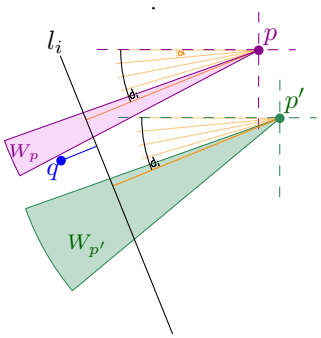


**Figure 9** Both $p$ and $p'$ belong to the $i$-th subset, and $l_i$ is the line perpendicular to $d_i$. $q$'s projection onto $l_i$ is below that of $p$, hence $q$ surely lies below the upper ray of $W_p$. Symmetrically, $q$ surely lies above the lower ray of $W_{p'}$.

In other words, case (f) can be handled, with some extra care, as a combination of cases (c) and (d).

Putting everything together, we obtain:

▶ **Theorem 3.** *BFS on $VG(P, r)$, in the undirected setup, can be performed in $O(n \log^{O(1)} n)$ randomized expected time.*

## 2.4 The corresponding bottleneck path problem

As before, we seek the minimum value $r^*$ for which there exists a path from $s$ to $t$ in $VG(P, r^*)$ (or a path of length at most $k$ in the bounded-hop version). This optimization problem can be solved using the shrink-and-bifurcate technique of [7, 11, 14], as above, using the BFS algorithm of Theorem 3 as the decision procedure. Here too, the critical values of $r$ are of two kinds: (i) pairwise distances between the points of $P$, and (ii) values at which some point $m^+(q)$ or $m^-(p)$ (or their symmetric counterparts) becomes co-vertical or co-horizontal with some point of $P$. Both cases are handled as before, first handling critical values of the second kind using standard parametric search, and then handling critical values of the first kind by the shrink-and-bifurcate technique. Omitting the obvious and easy adjustments, we obtain:

▶ **Theorem 4.** *The bottleneck path problem, as well as its bounded-hop (RSP) version, for undirected visibility graphs involving $n$ directional antennas in the plane, each with an opening angle at least $\alpha$, for some constant $\alpha$, can be solved in $O^*(n^{8/7})$ randomized expected time (where the $O^*(\cdot)$ factor also depends on $\alpha$).*

# 3 The case of terrains

Let $T$ be a 1.5-dimensional terrain, namely an $x$-monotone polygonal path, with $n$ vertices, let $P$ be a set of $m$ points on or above $T$, and let $r > 0$ be a real parameter. Consider the bounded-range visibility graph $VG(P, T, r)$, whose vertices are the points of $P$, and, for any $p, q \in P$, there is an edge between $p$ and $q$ iff they are mutually visible over $T$, meaning that the segment $pq$ lies fully above $T$, and $|pq|$ is at most $r$ (see Figure 10). The decision problem is to determine, for a given $r$, whether $VG(P, T, r)$ contains a path between two designated points $s, t \in P$, or, more generally, to perform BFS on $VG(P, T, r)$ from a given start point $s$. The optimization problem is the bottleneck path problem: Find the smallest value $r^*$ for which $VG(P, T, r^*)$ contains a path from $s$ to $t$. In the bounded-hop version, we

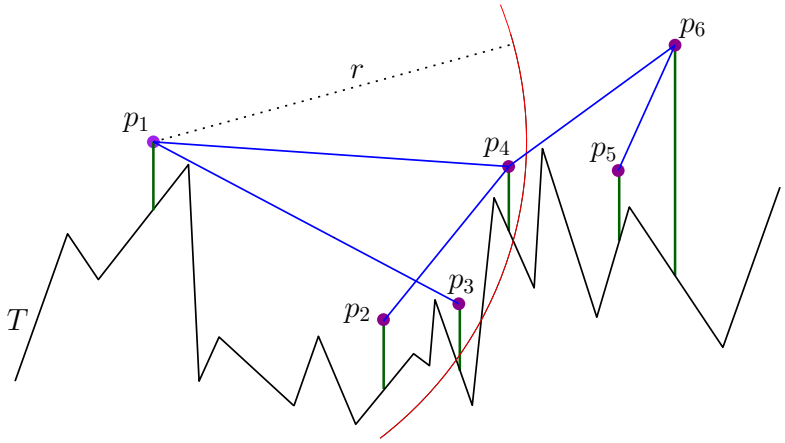


**Figure 10** The graph $VG(P, T, r)$. The points $p_1, p_6$ are mutually visible but $|p_1p_6| > r$, while $|p_1p_2| < r$ but $p_1, p_2$ are not mutually visible.

are also given an integer parameter $k < m$, and wish, as before, to determine whether there exists an $s$-$t$ path in $VG(P, T, r)$ with at most $k$ edges, or to find the minimum $r^*$ for which such a path exists.

A simpler version of this problem, without the bounded range restriction, has recently been studied in [16]. Notice that if we only wish to enforce the distance condition, we can use a dynamic Voronoi diagram of $P$, as before, at the cost of $O(\log^4 m)$ time per search and update. Similarly, if we only wish to enforce unconstrained visibility (namely, report points $p$ such that $qp$ lies above $T$), we can follow the query procedure given in [16] (see the full version for a brief review). The challenge, as before, is to enforce both constraints simultaneously and efficiently.

In the full version, we describe a multi-level data structure that achieves this goal at a polylogarithmic cost per query, using which we obtain the following two theorems; see the full version for details.

▶ **Theorem 5.** *BFS on $VG(P, T, r)$, for a 1.5-dimensional terrain $T$ with $n$ vertices and a set $P$ of $m$ points over $T$, can be performed in $O(n \log n + m \log^2 n + m \log^9 m \log n)$ randomized expected time.*

▶ **Theorem 6.** *The bottleneck path problem for bounded-range visibility graphs over a terrain, as well as its bounded-hop (RSP) version, can be solved in $O^*(n+m^{8/7})$ randomized expected time.*

## References


1 A. K. Abu-Affash, P. Carmi, M. J. Katz, and M. Segal. The Euclidean bottleneck steiner path problem and other applications of $(\alpha, \beta)$-pair decomposition. *Discrete & Computational Geometry*, 51:1–23, 2014. doi:10.1007/S00454-013-9550-9.

2 P. K. Agarwal, B. Aronov, M. Sharir, and S. Suri. Selecting distances in the plane. *Algorithmica*, 9:495–514, 1993. doi:10.1007/BF01187037.

3 P. K. Agarwal, H. Kaplan, M. J. Katz, and M. Sharir. Segment proximity graphs and nearest neighbor queries amid disjoint segments. *Algorithmica*, 88:55, 2026. doi:10.1007/s00453-026-01395-3.

4 P. K. Agarwal, M. J. Katz, and M. Sharir. On reverse shortest paths in geometric proximity graphs. *Computational Geometry*, 117:102053, 2024. doi:10.1016/j.comgeo.2023.102053.

5 R. Aschner and M. J. Katz. Bounded-angle spanning tree: Modeling networks with angular constraints. *Algorithmica*, 77(2):349–373, 2017. doi:10.1007/s00453-015-0076-9.

6 S. Ashur and M. J. Katz. A 4-approximation of the $(2\pi)/3$-mst. *Computational Geometry*, 108:101914, 2023. doi:10.1016/j.comgeo.2022.101914.

7 R. Ben Avraham, O. Filtser, H. Kaplan, M. J. Katz, and M. Sharir. The discrete and semicontinuous Fréchet distance with shortcuts via approximate distance counting and selection. *ACM Transactions on Algorithms*, 11(4):29:1–29:29, 2015. doi:10.1145/2700222.

8 A. Biniaz, P. Bose, A. Lubiw, and A. Maheshwari. Bounded-angle minimum spanning trees. *Algorithmica*, 84(1):150–175, 2022. doi:10.1007/s00453-021-00889-6.

9 A. Biniaz, M. Daliri, and A. H. Moradpour. A 10-approximation of the $\pi/2$-MST. *Journal of Computational Geometry*, 14(1):157–173, 2023. doi:10.20382/jocg.v14i1a6.

10 S. Cabello and M. Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry*, 48:360–367, 2015. doi:10.1016/j.comgeo.2014.12.003.

11 T. M. Chan and Z. Huang. Faster algorithms for reverse shortest path in unit-disk graphs and related geometric optimization problems: Improving the shrink-and-bifurcate technique. In *Proceedings of the 41st International Symposium on Computational Geometry (SoCG)*, pages 32:1–32:14, 2025. doi:10.4230/LIPIcs.SoCG.2025.32.

12 T. M. Chan and D. Skrepetos. All-pairs shortest paths in unit disk graphs in slightly subquadratic time. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC)*, pages 24:1–24:13, 2016. doi:10.4230/LIPIcs.ISAAC.2016.24.

13 M. de Berg and S. Cabello. An $O(n \log n)$ algorithm for single-source shortest paths in disk graphs. In *Proceedings of the 33rd Annual European Symposium on Algorithms (ESA)*, pages 81:1–81:15, 2025. doi:10.4230/LIPIcs.ESA.2025.81.

14 H. Kaplan, M. J. Katz, R. Saban, and M. Sharir. The unweighted and weighted reverse shortest path problem for disk graphs. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 67:1–67:14, 2023. doi:10.4230/LIPIcs.ESA.2023.67.

15 H. Kaplan, W. Mulzer, L. Roditty, P. Seiferth, and M. Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discrete & Computational Geometry*, 64:838–904, 2020. doi:10.1007/s00454-020-00243-7.

16 M. J. Katz, R. Saban, and M. Sharir. Near-linear algorithms for visibility graphs over a 1.5-dimensional terrain. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 77:1–77:17, 2024. doi:10.4230/LIPICS.ESA.2024.77.

17 K. Klost. An algorithmic framework for the single source shortest path problem with applications to disk graphs. *Computational Geometry*, 111:101979, 2023. doi:10.1016/j.comgeo.2022.101979.

18 C.-H. Liu. Nearly optimal planar $k$ nearest neighbors queries under general distance functions. *SIAM Journal on Computing*, 51(3):723–765, 2022. doi:10.1137/20M1388371.

19 M. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23:166–204, 1981. doi:10.1016/0022-0000(81)90012-X.

20 H. Wang and Y. Zhao. Reverse shortest path problem for unit-disk graphs. *Journal of Computational Geometry*, 14(1):14–47, 2023. doi:10.20382/JOCG.V14I1A2.