The homogeneous broadcast problem in narrow and wide strips I

Algorithmica (2019) 81:2934–2962
https://doi.org/10.1007/s00453-019-00567-8

The Homogeneous Broadcast Problem in Narrow and Wide Strips I: Algorithms

Mark de Berg[1] · **Hans L. Bodlaender**[2] · **Sándor Kisfaludi-Bak**[1]

Received: 12 December 2017 / Accepted: 23 March 2019 / Published online: 5 April 2019

Abstract

Let P be a set of nodes in a wireless network, where each node is modeled as a point in the plane, and let $s \in P$ be a given source node. Each node p can transmit information to all other nodes within unit distance, provided p is activated. The (homogeneous) broadcast problem is to activate a minimum number of nodes such that in the resulting directed communication graph, the source s can reach any other node. We study the complexity of the regular and the hop-bounded version of the problem (in the latter, s must be able to reach every node within a specified number of hops), with the restriction that all points lie inside a strip of width w. We describe several algorithms for both the regular and the hop-bounded versions, and show that both problems are solvable in polynomial time in strips of small constant width. These results complement the hardness results in a companion paper (de Berg et al. in Algorithmica, 2017).

Keywords Broadcast · Dominating set · Unit-disk graph · Range assignment

This research was supported by the Netherlands Organization for Scientific Research (NWO) under Project No. 024.002.003.

✉ Sándor Kisfaludi-Bak
s.kisfaludi.bak@tue.nl

Mark de Berg
m.t.d.berg@tue.nl

Hans L. Bodlaender
h.l.bodlaender@uu.nl

[1] Department of Mathematics and Computer Science, TU Eindhoven, PO Box 513, 5600 MB Eindhoven, The Netherlands

[2] Department of Computer Science, Utrecht University, Utrecht, The Netherlands

1 Introduction

Wireless networks give rise to a host of interesting algorithmic problems. In the traditional model of a wireless network each node is modeled as a point $p \in \mathbb{R}^2$, which is the center of a disk $\delta(p)$ whose radius equals the transmission range of p. Thus p can send a message to another node q if and only if $q \in \delta(p)$. Using a larger transmission radius may allow a node to transmit to more nodes, but it requires more power and is more expensive. This leads to so-called range-assignment problems, where the goal is to assign a transmission range to each node such that the resulting communication graph has desirable properties, while minimizing the cost of the assignment. We are interested in broadcast problems, where the desired property is that a given source node can reach any other node in the communication graph. Next, we define the problem more formally.

Let P be a set of n points in $\mathbb{R}^d$ and let $s \in P$ be a source node. A *range assignment* is a function $\rho : P \to \mathbb{R}_{\geqslant 0}$ that assigns a transmission range $\rho(p)$ to each point $p \in P$. Let $\mathcal{G}_\rho = (P, E_\rho)$ be the directed graph where $(p, q) \in E_\rho$ iff $|pq| \leqslant \rho(p)$. The function ρ is a *broadcast assignment* if every point $p \in P$ is reachable from s in $\mathcal{G}_\rho$. If every $p \in P$ is reachable within h hops, for a given parameter h, then ρ is an *h-hop broadcast assignment*. The (h-hop) broadcast problem is to find an (h-hop) broadcast assignment whose cost $\sum_{p \in P} \text{cost}(\rho(p))$ is minimized. Often the cost of assigning transmission radius x is defined as $\text{cost}(x) = x^\alpha$ for some constant α. In $\mathbb{R}^1$, both the basic broadcast problem and the h-hop version are solvable in $O(n^2)$ time [10]. In $\mathbb{R}^2$ the problem is NP-hard for any $\alpha > 1$ [9,20], and in $\mathbb{R}^3$ it is even APX-hard [20]. There are also several approximation algorithms [2,9]. For the 2-hop broadcast problem in $\mathbb{R}^2$ an $O(n^7)$ algorithm is known [3] and for any constant h there is a PTAS [3]. Interestingly, the complexity of the 3-hop broadcast problem is unknown.

An important special case of the broadcast problem is where we allow only two possible transmission ranges for the points, $\rho(p) = 1$ or $\rho(p) = 0$. In this case the exact cost function is irrelevant and the problem becomes to minimize the number of active points. This is called the *homogeneous broadcast problem* and it is the version we focus on. From now on, all mentions of broadcast and h-hop broadcast refer to the homogeneous setting. Observe that if $\rho(p) = 1$ then (p, q) is an edge in $\mathcal{G}_\rho$ if and only if the disks of radius $1/2$ centered at p and q intersect. Hence, if all points are active then $\mathcal{G}_\rho$ is the intersection graph of a set of congruent disks or, in other words, a *unit-disk graph*. Because of their relation to wireless networks, unit disk graphs have been studied extensively [1,24]. In addition, they are a fundamental graph class in computational geometry, and their study goes back several decades [8].

Let $\mathcal{D}$ be a set of congruent disks in the plane, and let $\mathcal{G}_\mathcal{D}$ be the unit disk graph induced by $\mathcal{D}$. A *broadcast tree* on $\mathcal{G}_\mathcal{D}$ is a rooted spanning tree of $\mathcal{G}_\mathcal{D}$. To send a message from the root to all other nodes, each internal node of the tree has to send the message to its children. Hence, the cost of broadcasting is related to the internal nodes in the broadcast tree. A cheapest broadcast tree corresponds to a minimum-size *connected dominating set* on $\mathcal{G}_\mathcal{D}$, that is, a minimum-size subset $\Delta \subset \mathcal{D}$ such that the subgraph induced by Δ is connected and each node in $\mathcal{G}_\mathcal{D}$ is either in Δ or a neighbor of a node in Δ. The broadcast problem is thus equivalent to the following:

given a unit disk graph $\mathcal{G}_\mathcal{D}$ with a designated source node s, compute a minimum-size connected dominated set $\Delta \subset \mathcal{D}$ such that $s \in \Delta$. The CONNECTED DOMINATING SET problem is recognized as a fundamental problem fro wireless network design, see the survey [31].

Given an algorithm for the broadcast problem, one can solve CONNECTED DOMINATING SET in unit disk graphs by running the algorithm n times, once for each possible source point. (In fact, we only need to run the algorithm $d_{\min}+1$ times, where $d_{\min}$ is the minimum degree of any vertex in the graph, since it suffices to try v and each of its neighbors as the source.) Consequently, hardness results for CONNECTED DOMINATING SET in unit disk graphs can be transferred to the broadcast problem, and algorithms for the broadcast problem can be transferred to CONNECTED DOMINATING SET in unit disk graphs at the cost of an extra linear factor in the running time. It is well known that DOMINATING SET and CONNECTED DOMINATING SET are NP-hard, even for planar graphs [21],and they remain NP-hard in unit disk graphs [25,28]. For any fixed d, both problems can be solved in $2^{O(n^{1-1/d})}$ time in unit balls graphs of $\mathbb{R}^d$, and even in more general intersection graphs [13]; this running time is tight under ETH. The parameterized complexity of DOMINATING SET in unit disk graphs has also been investigated: Marx [26] proved that DOMINATING SET in unit disk graphs is W[1]-hard when parameterized by the size of the dominating set, and De Berg et al. [15] showed that for most natural geometric intersection graphs (including unit disk graphs), DOMINATING SET is contained in W[1]. (The definition of W[1] and other parameterized complexity classes can be found in the book by Flum and Grohe [19].)

Knowing the existing hardness results for the broadcast problem, we set out to investigate the following questions. Is there a natural special case or parameterization admitting an efficient algorithm? Since the broadcast problem is polynomially solvable in $\mathbb{R}^1$, we study how the complexity of the problem changes as we go from the 1-dimensional problem to the 2-dimensional problem. To do this, we assume the points (that is, the disk centers) lie in a strip of width w, and we study how the problem complexity changes as we increase w. Such a restriction can be useful both from the applied and theoretical perspective: it may be useful to model ad-hoc networks along a street or highway [22], while concentrating on narrow strips is a natural first step to getting approximation algorithms using the well-known shifting technique [23,30].

An important threshold in the width of the strip is $w = \frac{\sqrt{3}}{2}$. This specific setup has been considered before for various problems [29,32]. Such *narrow strip unit disk graphs* are a subclass of co-comparability graphs, also known as incomparability graphs. Co-comparability graphs are graphs that can be obtained from a given partially ordered set $(P, \leqslant)$ by setting P as the vertex set and connecting pairs $a, b \in P$ if and only if they are not comparable. In case of a narrow strip $\mathbb{R} \times [0, \frac{\sqrt{3}}{2}]$ one can obtain a partial ordering on P by setting $u \leqslant v$ if and only if u and v are not connected in the unit disk graph and the x-coordinate of u is less than the x-coordinate of v [29]. It is routine to check that this relation is well-defined and transitive and that the co-comparability graph defined by $(P, \leqslant)$ is exactly the unit disk graph of P.

Our Contributions Our first result is an algorithm for broadcasting in narrow strips without a hop bound.

Theorem 1 *The broadcast problem inside a strip of width at most $\sqrt{3}/2$ can be solved in $O(n \log n)$ time.*

As remarked earlier, this result implies an $O(d_{\min} n \log n)$ algorithm for CONNECTED DOMINATING SET in narrow strip unit disk graphs, where $d_{\min}$ is the minimum degree in the graph. We can compare this to $O(mn)$, the running time that we get by applying the algorithm for the more general class of co-comparability graphs [5]. Since $m = \Omega(d_{\min} n)$, we get an almost linear speedup for the worst-case running time. If the graph is dense but has a constant-degree vertex, we even get a quadratic speedup, namely from $O(n^3)$ to $O(n \log n)$.

The key step toward getting this algorithm is a structural lemma stating that, except in some small-diameter instances that can be handled separately, there is always an optimal broadcast tree that induces a path in the underlying unit disk graph. In this case, the problem boils down to computing a shortest path from the source to some specific sets of points that are "far enough" to the right or to the left of the source.

The hop condition in the h-hop broadcast problem has not been studied yet for co-comparability graphs to our knowledge. This condition complicates the problem considerably. Our result here is as follows.

Theorem 2 *The h-hop broadcast problem inside a strip of width at most $\sqrt{3}/2$ can be solved in $O(n^6)$ time.*

The overall idea here is again to characterize optimal broadcasts. In each direction—that is, going to the right from the source, or going to the left—the optimal path without hop bound may already have at most h hops, in which case it is optimal. Otherwise, there may be points exactly $h + 1$ hops from the source that make it necessary to have multiple broadcasting points at h hops from the source. As a result, the broadcasting points form a tree-like structure, with the source as root and of depth h. To find such a tree, we use an adaptation of the Dreyfus–Wagner algorithm [17] for Steiner trees. For the case when the unit disk graph has a small diameter, our algorithm uses a subroutine for 2-hop broadcast, which may be of independent interest. Our subroutine is based on an algorithm by Ambühl et al. [3] for the non-homogeneous case, which runs in $O(n^7)$ time. We improve the running time of that algorithm for the homogeneous case to $O(n^4)$.

Finally, we investigate what happens for wider strips.

Theorem 3 *The broadcast problem and* CONNECTED DOMINATING SET *in unit disk graphs can be solved in $n^{O(w)}$ time on a strip of width w.*

The algorithm is a fairly straightforward dynamic programming using a sliding window of constant width.

As we show in a companion paper [11], this is likely best possible: we prove a matching lower bound of $n^{\Omega(w)}$, conditional on the Exponential Time Hypothesis (ETH), and prove that the problem is W[1]-complete when parameterized by the width w. As we show in the companion paper, we cannot hope to get such an algorithm for the h-hop broadcast problem, unless $\mathsf{P} = \mathsf{NP}$.

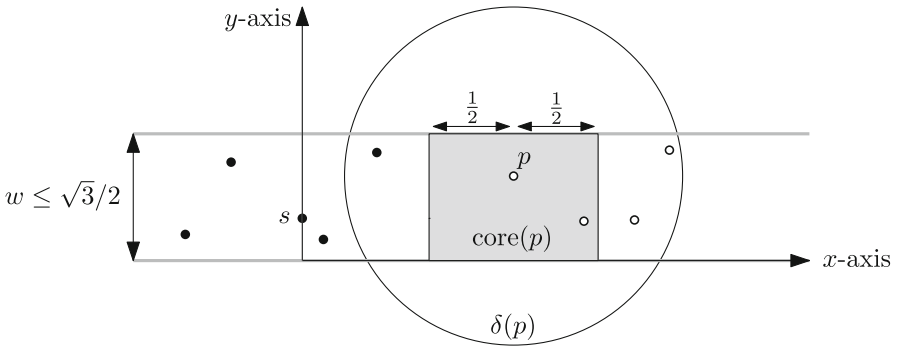


Fig. 1 A point p and its core. The set Q^+ of right-covering points—note that $p \in Q^+$, since all points to its right lie inside $\delta(p)$—is indicated by small circles, the other points are indicated by small (filled) disks

2 Algorithms for Broadcasting Inside a Narrow Strip

In this section we present polynomial algorithms (both for broadcast and for h-hop broadcast) for inputs that lie inside a strip $\mathcal{S} := \mathbb{R} \times [0, w]$, where $0 < w \leqslant \sqrt{3}/2$ is the width of the strip. Without loss of generality, we assume that the source lies on the y-axis. Define $\mathcal{S}_{\geqslant 0} := [0, \infty) \times [0, w]$ and $\mathcal{S}_{\leqslant 0} := (-\infty, 0] \times [0, w]$.

Let P be the set of input points. We define $x(p)$ and $y(p)$ to be the x- and y-coordinate of a point $p \in P$, respectively, and $\delta(p)$ to be the unit-radius disk centered at p. Let $\mathcal{G} = (P, E)$ be the graph with $(p, q) \in E$ iff $q \in \delta(p)$, and let $P' := P \setminus \delta(s)$ be the set of input points outside the source disk. We say that a point $p \in P$ is *left-covering* if $pp' \in E$ for all $p' \in P'$ with $x(p') < x(p)$; p is *right-covering* if $p'p \in E$ for all $p' \in P'$ with $x(p') > x(p)$; see Fig. 1. We denote the set of left-covering and right-covering points by Q^- and Q^+ respectively. Finally, the *core area* of a point p, denoted by $\text{core}(p)$, is $[x(p) - \frac{1}{2}, x(p) + \frac{1}{2}] \times [0, w]$. Note that $\text{core}(p) \subset \delta(p)$ because $w \leqslant \sqrt{3}/2$, i.e., the disk of p covers a part of the strip that has horizontal length at least one. This is a key property of strips of width at most $\sqrt{3}/2$, and will be used repeatedly.

We partition P into levels $L_0, L_1, \ldots L_t$, based on hop distance from s in $\mathcal{G}$. Thus $L_i := \{p \in P : d_{\mathcal{G}}(s, p) = i\}$, where $d_{\mathcal{G}}(s, p)$ denotes the hop-distance. Let L_i^- and L_i^+ denote the points of L_i with negative and nonnegative coordinates, respectively. We will use the following observation multiple times.

Observation 4 *Let $\mathcal{G} = (P, E)$ be a unit disk graph on a narrow strip $\mathcal{S}$.*

(i) *Let π be a path in $\mathcal{G}$ from a point $p \in P$ to a point $q \in P$. Then the region $[x(p) - \frac{1}{2}, x(q) + \frac{1}{2}] \times [0, w]$ is fully covered by the disks of the points in π.*

(ii) *The overlap of neighboring levels is at most $\frac{1}{2}$ in x-coordinates:* $\max\{x(p) | p \in L_{i-1}^+\} \leqslant \min\{x(q) | q \in L_i^+\} + \frac{1}{2}$ *for any $i > 0$ with $L_i^+ \neq \emptyset$; similarly,* $\min\{x(p) | p \in L_{i-1}^-\} \geqslant \max\{x(q) | q \in L_i^-\} - \frac{1}{2}$ *for any $i > 0$ with $L_i^- \neq \emptyset$.*

(iii) *Let p be an arbitrary point in L_i^+ for some $i > 0$. Then the disks of any path $\pi(s, p)$ cover all points in all levels $L_0 \cup L_1 \cup L_2^+ \cup \cdots \cup L_{i-1}^+$. A similar statement holds for points in L_i^-.*

Proof For (i), note that for any edge $(u, v) \in E$, we have that core(u) and core(v) intersect. Thus the union of the cores of the points of π is connected, and contains core(p) and core(q). Consequently, it covers $[x(p) - \frac{1}{2}, x(q) + \frac{1}{2}] \times [0, w]$.

To prove (ii), consider a point $p \in L_{i-1}^+$. Any shortest path $\pi(s, p)$ must contain a point $p' \in P$ with $0 \leqslant x(p') \leqslant x(p) - 1$. By (i) the disks of the points in the subpath $\pi(s, p')$ together cover the region $[-\frac{1}{2}, x(p') + \frac{1}{2}] \times [0, w]$. Hence, the level of any point in this region is at most level$(p') + 1 \leqslant i - 1$, and so a point $q \in L_i^+$ must have $x(q) > x(p') + \frac{1}{2} \geqslant x(p) - \frac{1}{2}$. Note that the term $\frac{1}{2}$ is tight, as can be seen by considering the set $P = \{(0, 0), (1, 0), (\frac{1}{2} + \varepsilon, \sqrt{3}/2)\}$.

Statement (iii) follows from (i) and (ii): the disks of $\pi(s, p)$ cover $\delta(s) \cup [-\frac{1}{2}, x(p) + \frac{1}{2}] \times [0, w]$, and $L_0 \cup L_1 \cup L_2^+ \cup \cdots \cup L_{i-1}^+$ is contained in this set. □

2.1 Minimum Broadcast Set in a Narrow Strip

A *broadcast set* is a point set $D \subseteq P$ that gives a feasible broadcast, i.e., a connected dominating set of $\mathcal{G}$ that contains s. Our task is to find a minimum broadcast set inside a narrow strip. Let $p, p' \in P$ be points with maximum and minimum x-coordinate, respectively. Obviously there must be paths from s to p and p' in $\mathcal{G}$ such that all points on these paths are active, except possibly p and p'. If p and p' are also active, then these paths alone give us a feasible broadcast set: by Observation 4(i), these paths cover all our input points. Instead of activating p and p', it is also enough to activate the points of a path that reaches Q^- and a path that reaches Q^+.

Lemma 5 *If there is a minimum broadcast set with an active point on L_2, then there is a minimum broadcast set consisting of the disks of a shortest path π^- from s to Q^- and a shortest path π^+ from s to Q^+. These two paths share s and they may or may not share their first point after s.*

Proof We begin by showing that there is a minimum broadcast that intersects both Q^+ and Q^-. Without loss of generality, we may assume that L_2^+ has an active point.

Claim *There is a minimum broadcast set D' containing a point in Q^+.*

Proof of Claim Let D be a minimum broadcast set. The active point in L_2^+ has a descendant leaf $a \in L_{\geqslant 2}^+$ in the broadcast tree (the tree one gets by performing breadth first search from s in the graph spanned by D). Note that $\delta(a)$ does not cover any points in $\mathcal{S}_{\leqslant 0} \backslash \delta(s)$, since $a \notin$ core(s) and core(s) has width 1.

Suppose that $D \cap Q^+ = \emptyset$. Since $a \notin Q^+$, there is a point $\bar{b}$ with a larger x-coordinate than a which is not covered by $\delta(a)$, but covered by another disk $\delta(b)$ for some $b \in D$. Similarly, there must be a point $\bar{a} \in \delta(a) \backslash \delta(b)$ with $x(\bar{a}) > x(b)$ (see Fig. 2 for an example). Since $\delta(b)$ covers core(b), we have $x(\bar{a}) > x(b) + \frac{1}{2}$, and similarly $x(\bar{b}) > x(a) + \frac{1}{2}$.

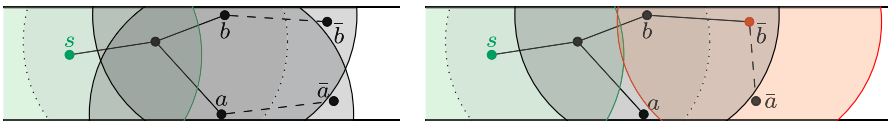


Fig. 2 A swap operation. The edges of the broadcast tree are solid lines

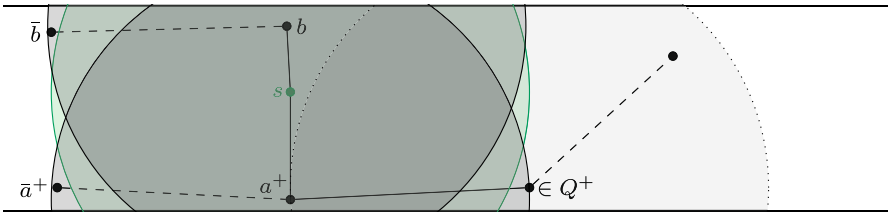


Fig. 3 If $D' \cap L_2^- = \emptyset$, we can still do swaps

Note that $x(\bar{b}) \leqslant x(b) + 1$, so $x(\bar{b}) - x(\bar{a}) < \frac{1}{2}$. The other direction yields $x(\bar{b}) - x(\bar{a}) > -\frac{1}{2}$, thus $\bar{a} \in \delta(\bar{b})$, or in other words, any point covered by $\delta(a)$ to the right of $\delta(b)$ can be covered by replacing $\delta(a)$ with $\delta(\bar{b})$. We call such a replacement a *swap operation*. This operation results in a new minimum broadcast set, because the size of the set remains the same, and no vertex can become disconnected from the source on either side: the right side remains connected along the broadcast tree, and the left is untouched since $\delta(a) \cap \mathcal{S}_{\leqslant 0} \subseteq \delta(s)$. Repeated swap operations lead to a minimum-size broadcast set D' that contains at least one point from Q^+. (The procedure terminates since the sum of the x-coordinates of the active points increases.) □

The resulting minimum broadcast set D' contains a path π^+ from s to Q^+. Let a^+ be the last point on π^+ that falls in L_1. Since $a^+ \in L_1$, we can remove any other vertices from π^+ that are between s and a^+, and step directly from s to a^+. That is, without loss of generality, we can assume that the first two points of π^+ are s and a^+. Let $q^+ = Q^+ \cap \pi^+$. By part (iii) of Observation 4, the disks around the points of π^+ cover all points with x coordinates between 0 and $x(q^+) + \frac{1}{2}$; and $q^+ \in Q^+$ implies that it covers all input points with x-coordinate higher than $x(q^+) + \frac{1}{2}$. Consequently, there are no active points in the right part outside this path—that is, no active points in $\mathcal{S}_{\geqslant 0} \setminus \big(\delta(s) \cup \pi^+\big)$—since those could be removed while maintaining the feasibility of the solution.

Claim *There is a minimum broadcast set D' containing a point in Q^+ and one in Q^-.*

Proof of Claim Consider a minimum broadcast set D' that has a point in Q^+, which exists by the previous claim. If there is a disk in $D' \cap L_2^-$, then we can reuse the previous argument for the other side, and get a broadcast set that contains a path π^- from s to Q^-. Otherwise, we need to be slightly more careful with our swap operations: we need to make sure not to remove a^+. If $a^+ \notin Q^-$, then we can again use the previous argument: it is possible to find another disk b, and

corresponding uniquely covered points $\bar{a}^+$ and $\bar{b}$ (see Fig. 3). Note that $b \in \delta(s)$ since we are in the case $D' \cap L_2^- = \emptyset$. We argue that b can be replaced with $\bar{a}^+$: removing b can not disconnect anything from s on either side, and $\delta(\bar{a}^+)$ covers all points covered by $\delta(b)$. Repeated swap operations lead to a minimum broadcast set D'' that contains points from both Q^+ and Q^-. □

Let π^- and a^- be defined analogously to how π^+ and a^+ were defined above. Note that a^+ and a^- might coincide. Since $\pi^+ \cup \pi^-$ is connected and covers all points, we have $D'' = \pi^+ \cup \pi^-$. To finish the proof, it remains to argue that we can take π^+ and π^- to be shortest paths to Q^+ and Q^-. Suppose π^+ is not a shortest path to Q^+. (The argument for π^- is similar.) Then we can replace $\pi^+ \cup \pi^-$ by $\overline{\pi}^+ \cup \pi^-$, where $\overline{\pi}^+$ is a shortest path from s to Q^+. Since π^+ and π^- share at most one point besides s, this replacement does not increase the size of the solution. □

Lemma 6 below fully characterizes optimal broadcast sets. To deal with the case where Lemma 5 does not apply, we need some more terminology. We say that the disk $\delta(q)$ of an active point q in a feasible broadcast set is *bidirectional* if there are two input points $p^- \in L_2^-$ and $p^+ \in L_2^+$ that are covered only by $\delta(q)$. See points p and p' in Fig. 5 for an example. Note that $q \in \text{core}(s)$, because $\text{core}(s) = [-\frac{1}{2}, \frac{1}{2}] \times [0, w]$ is covered by $\delta(s)$, and our bidirectional disk has to cover points both in $(-\infty, -\frac{1}{2}] \times [0, w]$ and $[\frac{1}{2}, \infty) \times [0, w]$. Active disks that are not the source disk and not bidirectional are called *monodirectional*.

Lemma 6 *For any input P that has a feasible broadcast set, there is a minimum broadcast set D that has one of the following structures.*

(i) Small*: $|D| \leqslant 2$.*
(ii) Path-like*: $|D| \geqslant 3$, and D consists of a shortest path π^- from s to Q^- and a shortest path π^+ from s to Q^+; π^+ and π^- share s and may or may not share their first point after s.*
(iii) Bidirectional*: $|D| = 3$, and D contains two bidirectional disk centers and s.*

Proof Let OPT be the size of a minimum broadcast set. First consider the case $\text{OPT} \geqslant 4$. By Lemma 5 it suffices to prove that there is an active point in L_2. If $L_3 \neq \emptyset$ this is trivially true, so assume that $L_3 = \emptyset$. Since $\text{OPT} \geqslant 4$, it follows that $L_2^+ \neq \emptyset$ otherwise activating the shortest path from s to the point with minimum x-coordinate is a feasible broadcast set of size at most 3. Similarly, $L_2^- \neq \emptyset$.

If $Q^+ \cap L_1 \neq \emptyset$, then there is a minimum broadcast set with an active point in L_2: we take s, a point from $Q^+ \cap L_1$, and a shortest path from s to the leftmost point (at most two more points). Thus we may assume that Q^+, and similarly, Q^- are disjoint from L_1.

Let $\{s, p_1, p_2, p_3\}$ be a subset of a minimum broadcast set. If $\delta(p_i)$ is monodirectional, then let $\bar{p}_i \in L_2$ be a point uniquely covered by p_i; suppose that $\bar{p}_i \in \mathcal{S}^+$ (the proof is the same for the left side). Since $p_i \notin Q^+$, there is a point $q \in L_1$ that uniquely covers another point $\bar{q} \in L_2$. We can swap p_i for $\bar{q}$ and get the desired outcome.

If all of $\delta(p_i)$ are bidirectional, then we can do a *double swap operation*: deactivate both $\delta(p_1)$ and $\delta(p_2)$, and activate $\delta(a^-)$ and $\delta(a^+)$, where a^- and a^+ are points

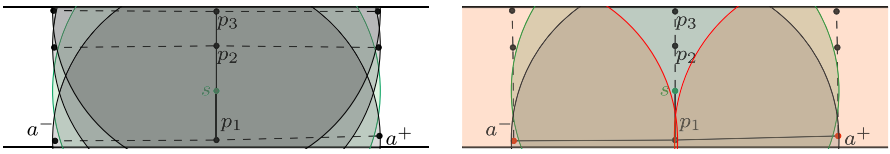
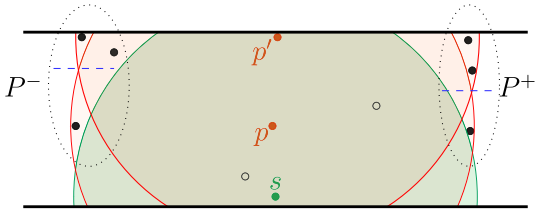


Fig. 4 A double swap operation. The edges of the broadcast tree are solid lines

Fig. 5 A bidirectional broadcast



uniquely covered by $\delta(p_3)$ on the left and right part of the strip (see Fig. 4). Note that $\delta(a^+)$ covers both $\mathcal{S}_{\geqslant 0} \cap (\delta(p_1) \backslash \delta(s))$ and $\mathcal{S}_{\geqslant 0} \cap (\delta(p_2) \backslash \delta(s))$, as we have seen this happen for regular swap operations in Lemma 5 – similarly, $\delta(a^-)$ covers both $\mathcal{S}_{\leqslant 0} \cap (\delta(p_1) \backslash \delta(s))$ and $\mathcal{S}_{\leqslant 0} \cap (\delta(p_2) \backslash \delta(s))$.

Therefore, the new broadcast set obtained after the double swap is feasible, and the size remains unchanged, so it is a minimum broadcast set. Notice that a single swap or double swap results in a minimum broadcast set that has an active point in L_2.

If the minimum broadcast set has size three, containing $\{\delta(s), \delta(p_1), \delta(p_2)\}$, then either both $\delta(p_1)$ and $\delta(p_2)$ are bidirectional, or at least one of them is monodirectional, so a single swap operation results in a minimum broadcast set with an active disk in L_2, so there is a path-like minimum broadcast set by Lemma 5. □

As it turns out, the bidirectional case is the most difficult one to compute efficiently. (It is similar to CONNECTED DOMINATING SET in co-comparability graphs, where the case of a connected dominating set of size at most 3 dominates the running time.)

Lemma 7 *In $O(n \log n)$ time we can find a bidirectional broadcast if it exists.*

Proof Let $P^- := \{u_1, u_2, \ldots, u_k\}$ be the set of points to the left of the source disk $\delta(s)$, where the points are sorted in increasing y-order with ties broken arbitrarily. Similarly, let $P^+ := \{v_1, v_2, \ldots, v_l\}$ be the set of points to the right of $\delta(s)$, again sorted in order of increasing y-coordinate. Define $P^-_{\leqslant i} := \{u_1, \ldots, u_i\}$, and define $P^-_{>i}$, and $P^+_{\leqslant i}$ and $P^+_{>i}$ analogously. Our algorithm is based on the following observation: There is a bidirectional solution if and only if there are indices i, j and points $p, p' \in \text{core}(s)$ such that $\delta(p)$ covers $P^-_{\leqslant i} \cup P^+_{\leqslant j}$ and $\delta(p')$ covers $P^-_{>i} \cup P^+_{>j}$; see Fig. 5.

Now for a point $p \in \text{core}(s)$, define $Z^-_{\leqslant}(p) := \max\{i : P^-_{\leqslant i} \subset \delta(p)\}$ and $Z^-_{>}(p) := \min\{i : P^-_{>i} \subset \delta(p)\}$, and $Z^+_{\leqslant}(p) := \max\{i : P^+_{\leqslant i} \subset \delta(p)\}$, and $Z^+_{>}(p) := \min\{i : P^+_{>i} \subset \delta(p)\}$. Then the observation above can be restated as:

There is a bidirectional solution if and only if there are points $p, p' \in \text{core}(s)$ such that $Z^-_{\leqslant}(p) \geqslant Z^-_{>}(p')$ and $Z^+_{\leqslant}(p) \geqslant Z^+_{>}(p')$.

Next we show how to find such a pair—if it exists—in $O(n \log n)$ time, once we have computed the values $Z^-_{\leqslant}(p)$, $Z^-_{>}(p)$, $Z^+_{\leqslant}(p)$, and $Z^+_{>}(p)$ for all points $p \in \delta(s)$. For each point $p \in \text{core}(s)$, define $\xi(p) := (Z^-_{\leqslant}(p), Z^+_{\leqslant}(p))$ and $\psi(p) := (Z^-_{>}(p), Z^+_{>}(p'))$. Thus we are looking for a pair of distinct points p, p' such that $\xi(p)$ lies to the north-east of $\psi(p')$. We can find such a pair (if it exists) with a plane-sweep algorithm that sweeps a vertical line from right to left over the plane and maintains the highest point $\xi(p)$ encountered so far.

It remains to show that these values can be computed in $O(n \log n)$ time.

Consider the computation of $Z^-_{\leqslant}(p)$; the other values can be computed similarly. Let $\mathcal{T}$ be a balanced binary tree whose leaves store the points from P^- in order of their y-coordinate. For a node v in $\mathcal{T}$, let $F(v) := \{\delta(u_i) : u_i \text{ is stored in the subtree rooted at } v\}$. We start by computing at each node v the intersection of the disks in $F(v)$. More precisely, for each v we compute the region $I(v) := \text{core}(s) \cap \bigcap F(v)$. Notice that $I(v)$ is y-monotone and convex, and each disk $\delta(u_i)$ contributes at most one arc to $\partial I(v)$. (Here $\partial I(v)$ refers to the boundary of $I(v)$ that falls inside $\mathcal{S}$.) Moreover, $I(v) = I(\text{left-child}(v)) \cap I(\text{right-child}(v))$. Hence, we can compute the regions $I(v)$ of all nodes v in $\mathcal{T}$ in $O(n \log n)$ time in total, in a bottom-up manner. Using the tree $\mathcal{T}$ we can now compute $Z^-_{\leqslant}(p)$ for any given $p \in \text{core}(s)$ by searching in $\mathcal{T}$, as follows. Suppose we arrive at a node v. If $p \in I(\text{left-child}(v))$, then descend to right-child(v), otherwise descend to left-child(v). The search stops when we reach a leaf, storing a point u_i. One easily verifies that if $p \in \delta(u_i)$ then $Z^-_{\leqslant}(p) = i$, otherwise $Z^-_{\leqslant}(p) = i - 1$.

Since $I(v)$ is a convex region, we can check if $p \in I(v)$ in $O(1)$ time if we can locate the position of p_y in the sorted list of y-coordinates of the vertices of $\partial I(v)$. We can locate p_y in this list in $O(\log n)$ time, leading to an overall query time of $O(\log^2 n)$. This can be improved to $O(\log n)$ using fractional cascading [7]. Note that the application of fractional cascading does not increase the preprocessing time of the data structure. We conclude that we can compute all values $Z^-_{\leqslant}(p)$ in $O(n \log n)$ time in total. □

In order to compute a minimum broadcast, we can first check for small and bidirectional solutions. To find path-like solutions, we first compute the sets Q^- and Q^+, and compute shortest paths starting from these sets back to the source disk. The path computation is very similar to the shortest path algorithm in unit disk graphs by Cabello and Jejčič [6].

Lemma 8 *Let P and Q be two point sets in $\mathbb{R}^2$. Then both $Q \cap \left(\bigcup_{p \in P} \delta(p)\right)$ and $Q \cap \left(\bigcap_{p \in P} \delta(p)\right)$ can be computed in $O((|P| + |Q|) \log |P|)$ time.*

Proof A point $q \in Q$ lies in $\bigcup_{p \in P} \delta(p)$ if and only if the distance from q to its nearest neighbor in P is at most 1. Hence we can compute $Q \cap \left(\bigcup_{p \in P} \delta(p)\right)$ by computing the Voronoi diagram of P, preprocessing it for point location, and performing a query with each $q \in Q$. This can be done in $O((|P| + |Q|) \log |P|)$ time in total [14,18]. To compute $Q \cap \left(\bigcap_{p \in P} \delta(p)\right)$ we proceed similarly, except that we use the farthest-point Voronoi diagram [14]. □

Lemma 9 *We can compute the sets Q^+ and Q^- in $O(n \log n)$ time.*

Proof We show how we can compute Q^+, the algorithm for Q^- is analogous. Let p be an input point with the highest x-coordinate. Notice that all input points in $[x(p) - \frac{1}{2}, x(p)] \times [0, w]$ belong to Q^+ since their core contains all points with higher coordinates. Points in $[x(p) - \frac{3}{2}, x(p) - 1) \times [0, w]$ cannot belong to Q^+, since they cannot cover p. It remains to find the points inside the region $\mathcal{R} = [x(p) - 1, x(p) - \frac{1}{2}) \times [0, w]$ that belong to Q^+. The core of a point in $\mathcal{R}$ covers $\mathcal{R}$, so it is sufficient to check whether any given point covers all points in $\mathcal{R}' = [x(p) - \frac{1}{2}, x(p)] \times [0, w]$. Thus we need to find the set $(\mathcal{R} \cap P) \cap \left(\bigcap_{p \in \mathcal{R}' \cap P} \delta(p)\right)$, which can be computed in $O(n \log n)$ time by Lemma 8. □

Proof (of Theorem 1) The algorithm can be stated as follows. It is best to read this pseudocode in parallel with the explanation and analysis below.

BROADCAST- IN- NARROW- STRIP(s, P)

1. Check if there is a small or bidirectional solution. If yes, report the solution and terminate.
2. Compute Q^+ using Lemma 9. Set $i := 1$, $Q_1^+ := Q^+$, and $P' := P \setminus Q_1^+$.
3. Repeat the following until $Q_i^+ \cap \delta(s) \neq \emptyset$ or $Q_i^+ = \emptyset$.
   (a) Set $i := i + 1$ and determine $T_i := \{t \in P' : x(t) \geqslant \min_{p \in Q_{i-1}^+} x(p) - 1\}$.
   (b) Compute $Q_i^+ := T_i \cap \left(\bigcup_{p \in Q_{i-1}^+} \delta(p)\right)$ using Lemma 8, and set $P' := P' \setminus Q_i^+$.
4. If $Q_i^+ = \emptyset$, return failure.
5. Compute Q^- using Lemma 9. Set $j := 1$, $Q_1^- := Q^-$, and $P' := P \setminus Q_1^-$.
6. Repeat the following until $Q_j^- \cap \delta(s) \neq \emptyset$ or $Q_j^- = \emptyset$.
   (a) Set $j := j + 1$ and determine $T_j := \{t \in P' : x(t) \leqslant \max_{p \in Q_{j-1}^-} x(p) + 1\}$.
   (b) Compute $Q_j^- := T_i \cap \left(\bigcup_{p \in Q_{j-1}^-} \delta(p)\right)$ using Lemma 8, and set $P' := P' \setminus Q_j^-$.
7. If $Q_j^- = \emptyset$, return failure.
8. If $Q_i^+ \cap Q_j^- = \emptyset$ then report a solution of size $i + j + 1$, namely the points of a shortest path from s to Q_i^+ and a shortest path from s to Q_j^-. Otherwise report a solution of size $i + j$: take an arbitrary point p in $Q_i^+ \cap Q_j^-$, and report s plus a shortest path from p to Q_i^+ and a shortest path from p to Q_j^-.

In order to execute step 1, we first check whether there is a minimum broadcast set of size one or two. This is very easy for size one: we just need to check whether the source disk covers every point or not in $O(n)$ time. For size two, we can compute the intersection of all disks centered outside $\delta(s)$, and check whether any input point in $\delta(s)$ falls in this intersection. This requires $O(n \log n)$ time by Lemma 8. Finally, we need to check whether there is a feasible minimum broadcast with the bidirectional structure. Lemma 7 shows that this is also possible in $O(n \log n)$ time.

In steps 2 and 3, we compute a shortest $s \to Q^+$ path backwards. We start from Q^+, and put the points into different sets Q_i^+ according to their hop distance to Q^+:

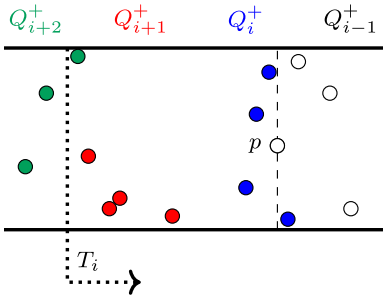


Fig. 6 The levels Q_i^+ computed by the algorithm

we put p into Q_i^+ if and only if the shortest path from p to Q^+ contains $i-1$ hops. Notice that in step 3 it is indeed sufficient to consider points from T_i, since a point from the level Q_i^+ must be at distance at most 1 from points of Q_{i-1}^+, so it has x-coordinate at least $\min_{p\in Q_{i-1}^+} x(p) - 1$.

If $Q_i^+ = \emptyset$, then there is no path from Q^+ to s—the graph is disconnected—so there is no feasible broadcast set. Otherwise, after the loop in step 3 terminates the shortest $s \to Q^+$ path has length exactly equal to the loop variable, i. Moreover, the set of possible second vertices on an $s \to Q^+$ path is $\delta(s) \cap Q_i^+$. The same can be said for the next two steps: the shortest $s \to Q^-$ path has length j, and the set of possible second vertices is $\delta(s) \cap Q_i^+$. In the final step, we check if $Q_i^+ \cap Q_j^-$ is empty or not. If it is empty, then by our previous observation, there are no shortest $s \to Q^+$ and $s \to Q^-$ paths that share their second vertex, so the two paths can only share s, resulting in a minimum broadcast set of size $i + j + 1$; otherwise, any point in $Q_i^+ \cap Q_j^-$ is suitable as a shared second point, resulting in a minimum broadcast set of size $i + j$.

It remains to argue that steps 2–8 require $O(n \log n)$ time. We know that a single iteration of the loop in step 3 takes $O\big((|Q_{i-1}^+| + |T_i|) \log |Q_{i-1}^+|\big)$ time by Lemma 8. We claim that $T_i \subseteq Q_i^+ \cup Q_{i+1}^+ \cup Q_{i+2}^+$, from which the bound on the running time follows. To prove the claim, let $p \in Q_{i-1}^+$ be a point with minimal x-coordinate (see Fig. 6). All points p' with $x(p') \geqslant x(p) - \frac{1}{2}$ are in $Q_{\leqslant i}^+$. Thus any point $p'' \in Q_{i+1}^+$ has $x(p'') < x(p) - \frac{1}{2}$. But then any point with x-coordinate at least $x(p) - 1$ also has x-coordinate at least $x(p'') - \frac{1}{2}$, which means it is in $Q_{\leqslant i+2}^+$. Thus both loops require $O(n \log n)$ time. Finally, we note that we can easily maintain some extra information in steps 2–7 so the shortest paths we need in step 8 can be reported in linear time. □

3 The 2-Hop Broadcast Problem

To compute a minimum-size broadcast set inside a narrow strip in the hop-bounded case, we will need a subroutine for the special case of two hops. For this we provide an algorithm that does not need that the points are inside a strip of width at most $\sqrt{3}/2$. Since this result is of independent interest, we provide it in a separate subsection.

Our algorithm is a modification of the $O(n^7)$ algorithm by Ambühl et al. [3]. Their algorithm works for the case where one can use different radii for the disks around

the points. For the homogeneous case that we consider (where a point is either active with unit radius or inactive) we obtain a better bound.

Theorem 10 *There is an algorithm that finds a minimum 2-hop broadcast set in $O(n^4)$ time.*

The proof of this theorem uses a slightly different approach, and the theorem itself is referenced only once; the reader may want to skip the rest of this section on the first reading.

Proof We start by testing if there is a solution consisting of a single disk (namely $\delta(s)$) or two disks ($\delta(s)$ and $\delta(p)$ for some $p \neq s$). This takes $O(n^2)$ time. If we do not find a solution of size one or two, we proceed as follows.

Let $Q := \{q_1, \ldots, q_m\}$ be the subset of points in P that are not covered by $\delta(s)$, where the points are numbered in counterclockwise order around s. We define $[i, j]$ to be the set of indices $\{i, \ldots, j\}$ if $i \leqslant j$, and we define $[i, j]$ to be the set of indices $\{i, \ldots, m, 1, \ldots, j\}$ if $i > j$. Furthermore, we define $Q[i, j]$ to be the set of points with indices in $[i, j]$. Let Δ be the set of disks (excluding the source disk $\delta(s)$) that may be useful in a minimum 2-hop broadcast. Obviously any point $p \in P$ with $\delta(p) \in \Delta$ must lie inside $\delta(s)$, because the broadcast is 2-hop. Moreover, $\delta(p)$ must contain at least one point $q_i \in Q$ to be useful.

We start by making sure that there is a feasible solution, so by checking that $Q \subseteq \bigcup \Delta$. The rest of the algorithm is a dynamic program, but we need several notations to describe it. The values $A[i, j]$ of our subproblems are defined as follows:

$A(i, j) :=$ the minimum number of disks from Δ needed to cover all points in $Q[i, j]$.

We will prove later that the size of an optimal broadcast set (not counting the source disk, and assuming that we need at least two disks in addition to the source disks) is given by

$$\text{OPT} = \min_{i,j} \big(A(i, j) + A(j + 1, i - 1)\big). \tag{1}$$

Define Δ_i to be the set of disks that can be used to cover a point $q_i \in Q$, that is,

$$\Delta_i := \{\delta \in \Delta : q_i \in \delta\}.$$

Let $next(i)$ be the first index in the sequence $[i, i-1]$ such that $Q[i, next(i)]$ cannot be covered by a single disk from Δ_i. (Such an index must exist since the solution size is at least three.) Furthermore, for a disk $\delta \in \Delta$, let $next(i, \delta)$ be the first index in $[i, i-1]$ such that $Q[i, next(i, \delta)]$ cannot be covered by δ. Thus $next(i) = \max_{\delta \in \Delta} next(i, \delta)$.

We now wish to set up a recurrence for $A(i, j)$. To this end, consider a disk $\delta \in \Delta$ and the point set $\delta \cap Q[i, j]$. The points in $\delta \cap Q[i, j]$ need not be consecutive in angular order around s: the disk δ may first cover a few points from $Q[i, j]$ (until $q_{next(i,\delta)-1}$), then there may be some points not covered, then it may cover some points again, and so on; see Fig. 7 where the angular ranges containing covered points are indicated in gray. We can thus define a set of maximal intervals that together form $\delta \cap Q[i, j]$:

$$\delta \cap Q[i, j] = Q[i, next(i, \delta) - 1] \cup Q[a_1, b_1] \cup Q[a_2, b_2] \cdots \cup Q[a_t, b_t].$$

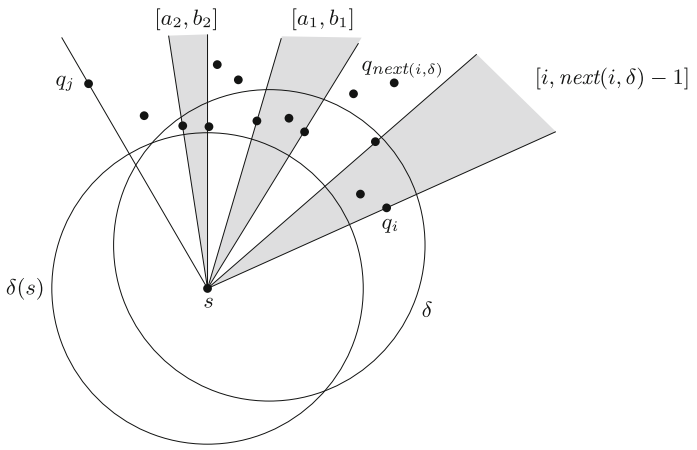


Fig. 7 Definition of the intervals $[a_i, b_i]$

Now define $\mathfrak{I}(i, j, \delta)$ as

$$\mathfrak{I}(i, j, \delta) := [a_1 - 1, b_1 + 1] \cup [a_2 - 1, b_2 + 1] \cdots \cup [a_t - 1, b_t + 1].$$

We claim that we now have the following recurrence:

$$A(i, j) = \begin{cases} 1 & \text{if } i = j \\ 1 + \min \left\{ A(next(i), j), \min\limits_{\substack{\delta \in \Delta_i \\ (a,b) \in \mathfrak{I}(i,j,\delta)}} (A(next(i, \delta), a) + A(b, j)) \right\} & \text{otherwise.} \end{cases} \tag{2}$$

We need to establish some key properties to prove the correctness of this recurrence. Let D be the set of active points in a minimum-size 2-hop broadcast. We call a disk $\delta(p)$ of an active point p an *active disk*. Let $\mathcal{U}(D) := \bigcup\{\delta(p) : p \in D\}$ be the union of the active disks. □

Observation 11 *The region $\mathcal{U}(D)$ is star-shaped with respect to the source point s, that is, for any point z in $\mathcal{U}$, the segment sz is inside $\mathcal{U}(D)$.*

Proof Let $p \in D$ be a point such that $z \in \delta(p)$. Suppose for contradiction that there is a point $t \in sz$ that lies outside $\mathcal{U}(D)$, and let ℓ be the perpendicular bisector of tz. Since $t \notin \delta(p)$, point p lies on the same side of ℓ as z. Note that since $t \notin \delta(s)$, the disk $\delta(s)$ is entirely covered by the other half plane of ℓ. Thus $p \notin \delta(s)$, which is a contradiction since in a 2-hop broadcast set we have $D \subset \delta(s)$. □

Let $\partial\mathcal{U}(D)$ be the boundary of $\mathcal{U}(D)$. By the previous observation, $\partial\mathcal{U}(D)$ is connected for 2-hop broadcast sets. Note that a point $q \in Q$ can be covered by multiple active disks. We will assign a unique point $pred(q) \in D$ whose disk covers q to each

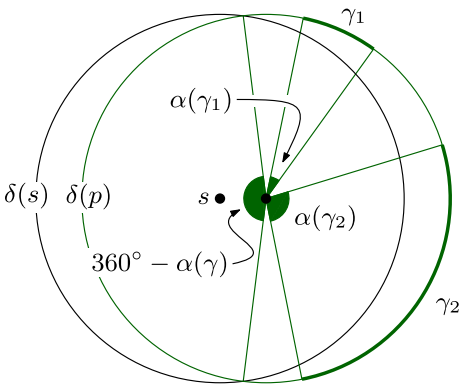


Fig. 8 Illustration for the proof of Lemma 13

$q \in Q$, as follows. We call $pred(q)$ the *predecessor* of q (in the given solution D) because $pred(q)$ can be thought of as the predecessor of q in a broadcast tree induced by D. Let $ray(q)$ be the ray emanating from s and passing through q, and consider the point z where $ray(q)$ exits $\mathcal{U}(D)$. Then we define $pred(q)$ to be the point that is the center of the active disk δ on whose boundary z lies (with ties broken arbitrarily, but consistently).

Recall that the points in Q are numbered in angular order around s, and consider the circular sequence $\sigma(D) := \langle pred(q_1), \ldots, pred(q_m)\rangle$. We modify $\sigma(D)$ by replacing any consecutive subsequence consisting of the same point by a single occurrence of that point. For example, we would modify $\langle p, p, p, q, q, p, p, r, r, r, p\rangle$ to obtain $\langle p, q, p, r, p\rangle$.

Observation 12 *In a 2-hop broadcast set D, the boundary sequence $\sigma(D)$ has no cyclic subsequence $\ldots p \ldots p' \ldots p \ldots p'$ with $p \neq p'$.*

Proof Between two adjacent occurrences of p and p' on the boundary, there must be an intersection between p and p'. Since there can be at most two intersections between two circles, the sequence $\ldots p \ldots p' \ldots p \ldots p'$ cannot occur in σ. □

Lemma 13 *In a 2-hop broadcast set D, any point $p \in D$ can appear in $\sigma(D)$ at most twice.*

Proof Consider the part of the boundary $\partial\delta(p)$ lying outside the source disk $\delta(s)$. This boundary part, which we denote by γ, can be partitioned into arcs where $\partial\delta(p)$ defines $\partial\mathcal{U}(D)$ and arcs where it does not. Assume for a contradiction that there are three arcs where $\partial\delta(p)$ defines $\partial\mathcal{U}(D)$—obviously this is necessary for p to appear three times in $\sigma(D)$. Then there must be two arcs, γ_1 and γ_2, where $\partial\delta(p)$ does not define $\partial\mathcal{U}(D)$ and such that γ_1 and γ_2 lie fully in the interior of γ. Let $\alpha(\gamma)$ denote the opening angle of the cone with apex p defined by γ, and define $\alpha(\gamma_1)$ and $\alpha(\gamma_2)$ similarly; see Fig. 8. It is easy to see that $\alpha(\gamma) \leqslant 240^\circ$. Since γ_1 and γ_2 do not cover γ completely then one of them, say γ_1, must be less than 120°. We will show that this leads to a contradiction, thus proving the lemma.

Let $\delta(p')$ be a disk covering (part of) γ_1. Since $\delta(p')$ covers less than 120° of γ_1, its center p' must lie outside $\delta(p)$. On the other hand, p' must lie inside $\delta(s)$, since we

have a 2-hop broadcast and $p' \in D$. Now observe that p' lies on the ray ρ starting at p that goes through the midpoint of the arc $\gamma_1 \cap \delta(p')$. This is a contradiction because ρ is disjoint from $\delta(s) \backslash \delta(p)$. □

We are now ready prove the correctness of our algorithm.

First consider Eq. (1). It is clear that

$$\text{OPT} \leqslant \min_{i,j} (A(i, j) + A(j + 1, i - 1))$$

since the union of the best covering of $Q[i, j]$ and $Q[j+1, i-1]$ is a feasible covering.

To prove the reverse, let D be a minimum-size 2-hop broadcast set. Suppose some point, p, appears only once in $\sigma(D)$. Let i, j be such that $\{q \in Q : pred(q) = p\} = Q[i, j]$. Then $A(i, j) = 1$ and there is a covering of $Q[j + 1, i - 1]$ with $|D| - 1$ disks. Hence, $\min_{i,j} (A(i, j) + A(j + 1, i - 1)) \leqslant \text{OPT}$ in this case. If all points appear twice in $\sigma(D)$ then we can argue as follows. Consider a point $p \in D$, and let i_1, j_1 and i_2, j_2 be such that $\{q \in Q : pred(q) = p\} = Q[i_1, j_1] \cup Q[i_2, j_2]$. Then the set of disks used by D in the covering of $Q[i_1, j_2]$ is disjoint from the set of disks used by D in the covering of $Q[j_2 + 1, i_1 - 1]$ by Observation 12. Hence, $(A(i_1, j_2) + A(j_2 + 1, i_1 - 1)) \leqslant \text{OPT}$.

Next, we prove that the recursive formula (2) holds. We prove this by induction on the length of $[i, j]$. If $i = j$, then $A(i, j) = 1$ is correct since our initial feasibility check implies that there is at least one disk $\delta \in \Delta$ that can cover q_i. Now consider the case $i \neq j$. First we note that

$$A(i, j) \leqslant 1 + \min \left\{ A(next(i), j), \min_{\substack{\delta \in \Delta_i \\ (a,b) \in \mathfrak{I}(i,j,\delta)}} (A(next(i, \delta), a) + A(b, j)) \right\}.$$

Indeed, there is a disk covering $Q[i, next(i) - 1]$ by definition of $next(i)$ and we can cover $Q[next(i), j]$ by $A(next(i), j)$ disks by induction. Similarly, the definition of $\mathfrak{I}(i, j, \delta)$ implies that any disk $\delta \in \Delta_i$ covers $Q[i, next(i, \delta) - 1]$ and $Q[a + 1, b - 1]$. By induction we can thus cover $Q[i, j]$ by $1 + A(next(i, \delta), a) + A(b, j)$ disks.

To prove the reverse, let D be a minimum-size 2-hop broadcast for $Q[i, j]$ and let $p := pred(q_i)$. If p appears in the covering of $Q[i, j]$ only once, then $A(i, j) = 1 + A(next(i), j)$. Otherwise p appears twice by Lemma 13. Let q_a be the last point before the second appearance of p in $\sigma(D)$, and let q_b be the first point after the second appearance of p in σ. By Observation 12, the coverings of $Q[next(i, \delta), a]$ and $Q[b, j]$ are disjoint in D. Hence, $1 + (A(next(i, \delta), a) + A(b, j)) \leqslant |D|$. We conclude that

$$A(i, j) \geqslant 1 + \min \left\{ A(next(i), j), \min_{\substack{\delta \in \Delta_i \\ (a,b) \in \mathfrak{I}(i,j,\delta)}} (A(next(i, \delta), a) + A(b, j)) \right\}.$$

It remains to analyze the running time. The algorithm works by first computing $next(i)$, $next(i, \delta)$ and $\mathfrak{I}(i, j, \delta)$ for each i, j and $\delta \in \Delta$. This can easily be done in $O(n^4)$ time. Running the dynamic program using the recursive formula (2) then takes

$O(n^4)$ time, as we have $O(n^2)$ entries $A(i, j)$ that each can be computed in $O(n^2)$ time. Finally, computing the optimal solution using Eq. (1) takes $O(n^2)$ time. Hence, the overall time requirement is $O(n^4)$, while the space required is $O(n^2)$. Computing an optimal solution itself, rather than just the value of OPT, can be done in a standard manner, without increasing the time or space bounds. □

4 Minimum-Size *h*-hop Broadcast in a Narrow Strip

In the hop-bounded version of the problem we are given P and a parameter h, and we want to compute a minimum broadcast set D such that every point $p \in P$ can be reached in at most h hops from s. In other words, for any $p \in P$, there must be a path in $\mathcal{G}$ from s to p of length at most h, all of whose vertices, except possibly p itself, are in D. We start by investigating the structure of optimal solutions in this setting, which can be very different from the non-hop-bounded setting.

As before, we partition P into levels L_i according to the hop distance from s in the graph $\mathcal{G}$. Recall that L_i^+ and L_i^- are the subsets of points at level i with positive and nonnegative x-coordinates, respectively. Let L_t be the highest non-empty level. If $t > h$ then clearly there is no feasible solution.

If $t < h$ then we can safely use our solution for the non-hop-bounded case, because the non-hop-bounded algorithm gives a solution which contains a path with at most $t + 1$ hops to any point in P. This follows from the structure of the solution; see Lemma 6. (Note that it is possible that the solution given by this algorithm requires $t + 1$ hops to some point, namely, if $Q^+ \cup Q^- \subseteq L_t$.) With the $t < h$ case handled by the non-hop-bounded algorithm, we are only concerned with the case $t = h$.

We deal with *one-sided* inputs first, where the source is the leftmost input point. Let $\mathcal{G}^*$ be the directed graph obtained by deleting edges connecting points inside the same level of $\mathcal{G}$, and orienting all remaining edges from lower to higher levels. A *Steiner arborescence* of $\mathcal{G}^*$ for the terminal set L_h is a directed tree rooted at s that contains a (directed) path π_p from s to p for each $p \in L_h$. From now on, whenever we speak of *arborescence* we refer to a Steiner arborescence in $\mathcal{G}^*$ for terminal set L_h. We define the *size* of an arborescence to be the number of internal nodes of the arborescence. Note that the leaves in a minimum-size arborescence are exactly the points in L_h: these points must be in the arborescence by definition, they must be leaves since they have out-degree zero in $\mathcal{G}^*$, and leaves that are not in L_h can be removed.

Remark 1 In the minimum Steiner Set problem, we are given a graph G and a vertex subset T of terminals, and the goal is to find a minimum-size vertex subset S such that $T \cup S$ induces a connected subgraph. This problem has a polynomial algorithm in co-comparability graphs [5], and therefore in narrow strip unit disk graphs. However, the broadcast set given by a solution does not fit our hop bound requirements. Hence, we have to work with a different graph (e.g. the edges within each level L_i have been removed), and this modified graph is not necessarily a co-comparability graph.

Lemma 14 below states that either we have a path-like solution—for the one-sided case a path-like solution is a shortest $s \to Q^+$ path— or any minimum-size arborescence defines a minimum-size broadcast set. The latter solution is obtained by

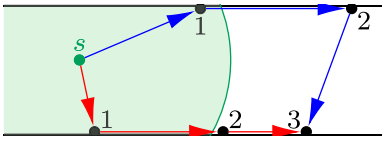


Fig. 9 Two different arborescences, with vertices labeled with their level. The red arborescence does not define a feasible broadcast for $h = 3$, since it would take four hops to reach the top right node (Color figure online)

activating all non-leaf nodes of the arborescence. We denote the broadcast set obtained from an arborescence A by D_A.

Lemma 14 *Any minimum-size Steiner arborescence for the terminal set L_h defines a minimum broadcast set, or there is a path-like minimum broadcast set.*

Proof Let A be a minimum Steiner arborescence for the terminal set L_h. Suppose that the broadcast set D_A defined by the internal vertices of A is not an h-hop broadcast set. (If it is, it must also be minimum and we are done.) By the properties of the arborescence every point in D_A can be reached in at most $h-1$ hops. Hence, if there is a point $p \in P$ that cannot be reached within h hops via D_A then p cannot be reached at all via D_A. Let i be such that $p \in L_i$. Since $L_h \subset A$, we know that $i < h$. Take any path from s to any point in L_{h-1} inside the arborescence. By Observation 4(iii), this path covers all lower levels. Hence, $i \geqslant h-2$, which implies $p \in L_{h-1}$.

Without loss of generality, suppose that p has the highest x-coordinate among points not covered by A. Let q be the point in P with the largest x-coordinate. If $q \in L_{\leqslant h-1}$, then a shortest $s \to q$ path is a feasible broadcast set of size at most $|A|$ that is path-like. Therefore, we only need to deal with the case $q \in L_h$. Let $p' \in A$ be an internal vertex of the arborescence whose disk covers q. The arborescence contains an $s \to p'$ path, which, by Observation 4(i), covers everything with x-coordinate up to $x(p') + \frac{1}{2}$. Since $p \notin \delta(p')$, we have $x(p) > x(p') + \frac{1}{2} \geqslant x(q) - \frac{1}{2}$. Since q has the maximum x coordinate, Observation 4(i) shows that the disks of a shortest $s \to p$ path form a feasible broadcast set, which is a path-like solution. □

Notice that a path-like solution also corresponds to an arborescence. However, it can happen that there are minimum-size arborescences that do not define a feasible broadcast; see Fig. 9. Lemma 14 implies that if this happens, then there must be an optimal path-like solution. The lemma also implies that for non-path-like solutions we can use the Dreyfus–Wagner dynamic-programming algorithm to compute a minimum Steiner tree [17], and obtain an optimal solution from this tree.[1] Unfortunately the running time is exponential in the number of terminals, which is $|L_h|$ in our case. However, our setup has some special properties that we can use to get a polynomial algorithm.

We define an arborescence A to be *nice* if the following holds. For any two arcs uu' and vv' of A that go between the same two levels, with $u \neq v$, we have: $y(u') <$

[1] The Dreyfus–Wagner algorithm minimizes the number of edges in the arborescence. In our setting the number of edges equals the number of internal nodes plus $|L_h| - 1$, so this also minimizes the number of internal nodes.

$y(v') \Rightarrow y(u) < y(v)$. Intuitively, a nice arborescence is one consisting of paths that can be ordered vertically in a consistent manner, see the left of Fig. 10. We define an arborescence A to be *compatible* with a broadcast set D if $D = D_A$. Note that there can be multiple arborescences—that is, arborescences with the same node set but different edge sets—compatible with a given broadcast set D.

Observation 15 *In a minimum broadcast set on the strip, the difference in x-coordinates between active points from a given level L_i ($i \leqslant h-1$) is at most $\frac{1}{2}$.*

Proof Let p and q be active points from L_i, and suppose for contradiction that $x(p) > x(q) + \frac{1}{2}$. By Observation 4(i), all points to the left of p are covered by the active points, so we only need to show that there are no points in L_{i+1} whose hop distance becomes longer by removing $\delta(q)$ from the solution. Indeed, consider a point $v \in L_{i+1} \cap (\delta(q) \backslash \delta(p))$. Since $\delta(q) \backslash \delta(p)$ lies to the left of p, $x(v) < x(p)$. So v has a path of at most $i+1$ hops. Hence we still have a feasible solution after removing $\delta(q)$, which contradicts the optimality of the original solution. □

Lemma 16 *Let $p \in L_i$ be a point in an optimal broadcast set D. Then there is a path of length i from s to p in $\mathcal{G}[D]$, the graph induced by D.*

Proof We say that a vertex $p \in L_i \cap D$ is *bad* if the shortest path in $\mathcal{G}[D]$ has more than i hops. Let p be a bad vertex of highest level among the bad vertices. If $i = h$, then the broadcast set is infeasible, thus $i \leqslant h-1$. If $p \in L_{h-1}$, then the shortest $s \to p$ path in $\mathcal{G}[D]$ must have length h, consequently, p cannot be used in an h-hop path to any other point. Therefore, p can be deactivated. (Note that p itself remains covered since it was reachable in the first place.)

If p is on a lower level, then let π_q be a shortest path in $\mathcal{G}[D]$ going to the last level, and let $q \in \pi_q \cap L_{h-1}$. Let π_p be the shortest $s \to p$ path in $\mathcal{G}[D]$. Note that π_q covers all lower levels $L_{\leqslant h-2}$ using at most h hops. Since i is the highest level with a bad point, all points $v \in D \cap L_{\geqslant i+1}$ have a shortest path in $\mathcal{G}[D]$, and such a path cannot pass through p.

Since p is a necessary point in this broadcast, and it is already covered by the disks of π_q in at most h hops, there must be a point p' to which all covering paths of length at most h pass through p. Since all points of L_h are covered by $D \cap L_{h-1}$ and $L_{\leqslant h-2}$ is covered by π_q, the level of p' has to be $h-1$. A covering path to p' has only bad vertices after p, so its point in L_{h-2} is bad. By the choice of p, we have $p \in L_{h-2}$, and since p' is reached in exactly h hops, it also follows that $p' \in \delta(p)$.

Note that p' cannot be to the left of $\delta(q)$, since then π_q would cover it in at most h hops; therefore, $x(p') > x(q) + \frac{1}{2}$. It follows that $x(p) \geqslant x(q) - \frac{1}{2}$, so $\delta(p)$ covers q. Since q is an arbitrary point in $D \cap L_{h-1}$, we have $D \cap L_{h-1} \subseteq \delta(p)$. Let D' be the broadcast obtained by replacing $D \cap L_{\leqslant h-2}$ with a shortest $s \to p$ path π'_p. We claim that $D' = \pi'_p \cup (D \cap L_{h-1})$ is a feasible broadcast: it covers L_h since points of L_h could only be covered by $D \cap L_{h-1}$, and it is easy to check that all points are covered in at most h hops. We arrived at a contradiction since D' is smaller than $\pi_p \cup (D \cap L_{h-1}) \subseteq D$. □

Lemma 17 *Every optimal broadcast set D has a nice compatible arborescence.*

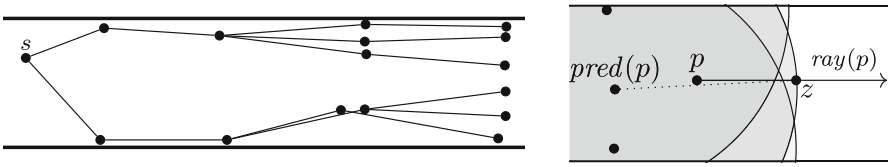


Fig. 10 Left: A nice Steiner arborescence. Note that arc crossings are possible. Right: Defining the *pred* function

Proof To find a nice compatible arborescence we will associate a unique arborescence with D. To this end, we define for each $p \in (D \cup L_h)\backslash\{s\}$ a unique predecessor $pred(p)$, as follows. Let ∂_i^* be the boundary of $\bigcup\{\delta(p)|p \in L_i \cap D\}$. It follows from Observation 15 that the two lines bounding the strip $\mathcal{S}$ cut ∂_i^* into four parts: a top and a bottom part that lie outside the strip, and a left and a right part that lie inside the strip. Let ∂_i be the part on the right inside the strip.

We then define the function $pred : (D \cup L_h)\backslash\{s\} \to D$ the following way. Consider a point $p \in (D \cup L_h)\backslash\{s\}$ and let i be its level. Let $ray(p)$ be the horizontal ray emanating from p to the right; see the right of Fig. 10. Let $pred(p)$ be the center of the disk that contains the point $z := ray(p) \cap \partial_{i-1}$. If there are multiple such disks, we can break ties by choosing $pred(p)$ to be the point with the highest y-coordinate in $L_i \cap D$ whose disk passes through z. It follows from Observation 4(iii) that $ray(q)$ cannot enter any disk from level $i-1$. Since any point $p \in D \cap L_h$ is contained in a disk from the preceding level, the point $pred(p)$ is well-defined for these points. The edges $pred(p)p$ for $p \in D \cap L_h$ thus define an arborescence. We can prove that it is nice by showing that the y-order of the points in a level L_i corresponds to the vertical order in which the boundaries of their disks appear on $\bigcup\{\delta(p) : p \in L_i \cap D\}$.

Let A be the directed graph defined by the edges $pred(p)p$ for each $p \in (D \cup L_h)\backslash\{s\}$. We show that A is a nice arborescence. By definition of the *pred*-function, each edge is between points at distance at most 1 that are in subsequent levels. Hence, the edges we add define an arborescence A on $\mathcal{G}^*$ with terminal set L_h. It remains to prove that A is nice.

Consider the edges of A going between points in L_{i-1} and points in L_i. By drawing horizontal lines through each of the breakpoints of ∂_{i-1}, the strip $\mathcal{S}$ is partitioned into horizontal sub-strips, such that two points from L_i are assigned the same predecessor iff they lie in the same sub-strip. Number the sub-strips $\mathcal{S}_1, \mathcal{S}_2, \ldots$ in vertical order, with $\mathcal{S}_1$ being the bottommost sub-strip. Let $u_j \in D \cap L_{i-1}$ be the point that is the predecessor of the points in the sub-strip $\mathcal{S}_j$. To show that A is nice, it is sufficient to demonstrate that the sequence $u_1, u_2, \ldots$ is ordered by the y-coordinates of the points.

Suppose for a contradiction that this is not the case. Then there are points u_j and u_{j+1} such that $y(u_j) > y(u_{j+1})$. Let z be the breakpoint on ∂_{i-1} between the arcs defined by $\delta(u_j)$ and $\delta(u_{j+1})$. Since z is in the right half circle of both $\delta(u_j)$ and $\delta(u_{j+1})$, we have $\max\{x(u_j), x(u_{j+1})\} < x(z)$. Since $|u_j z| = |u_{j+1} z| = 1$, the point z lies on the perpendicular bisector of $u_j u_{j+1}$ to the right of u_j and u_{j+1}. Since $y(u_j) > y(u_{j+1})$, the outer circle below the bisector is $\delta(u_{j+1})$ and the outer circle above the bisector is $\delta(u_j)$. This contradicts the ordering of the sub-strips. □

Let $q_1, q_2, \ldots, q_m$ be the points of L_h in increasing y-order. The crucial property of a nice arborescence is that the descendant leaves of a point p in the arborescence form an interval of $q_1, q_2, \ldots, q_m$. Using the above lemmas, we can adapt the Dreyfus–Wagner algorithm and get the following theorem.

Theorem 18 *The one-sided h-hop broadcast problem inside a strip of width at most $\sqrt{3}/2$ can be solved in $O(n^4)$ time.*

Proof By our lemmas, we know that our solution can be categorized as path-like or as arborescence-based. We compute the best path-like solution by invoking the second part of our narrow strip broadcast algorithm, which runs in $O(n \log n)$ time. The output of this algorithm is a path with t or $t+1$ hops (where t is the number of levels); thus, it is a minimum h-hop broadcast set if $t < h$, or if $t = h$ and the path has length h. Otherwise there is no path-like h-hop broadcast set, so an arborescence defines a minimum h-hop broadcast set by Lemma 14. By Lemma 17, it is sufficient to look for a nice Steiner arborescence, and take the broadcast set defined by it.

The algorithm to find a nice Steiner arborescence is based on dynamic programming. A subproblem is defined by a point $p \in P$ and an interval of the last level (that is, an interval of the sequence $q_1, q_2, \ldots, q_m$, the points of L_h ordered by y-coordinates). The solution of the subproblem $M(p, [i, j])$, for $1 \leqslant i \leqslant j \leqslant m$, is the minimum number of internal vertices in a nice arborescence which is rooted at p and contains $q_i, q_{i+1}, \ldots, q_j$ as leaves. Recall that $d_{\mathcal{G}^*}(p, q)$ denotes the hop distance function in $\mathcal{G}^*$, where $d_{\mathcal{G}^*}(p, q) = \infty$ if there is no path from p to q. We claim that the following recursion holds:

$$M(p, [i, j]) = \begin{cases} d_{\mathcal{G}^*}(p, q_i) - 1 & \text{if } i = j, \\ \min\left(1 + \min_{\substack{p' \in P \cap \delta(p) \\ p' \neq p}} M(p', [i, j]),\ \min_{i \leq t \leq j-1}\big(M(p, [i, t]) + M(p, [t+1, j])\big)\right) & \text{if } i < j. \end{cases} \tag{3}$$

The number of subproblems is $O(n^3)$, each of them requires computing the minimum of at most $O(n)$ values. This results in an algorithm that runs in $O(n^4)$ time. The minimum broadcast set size is $M(s, [1, m])$; if we keep track of a representing arborescence for each subproblem, we can also return a minimum broadcast set without any extra runtime cost.

To prove correctness, we need to show that Eq. (3) is correct. The base case, $i = j$, is obviously correct, so now assume $i < j$. It is easily checked that $M(p, [i, i])$ is at most the right-hand side of the equation. For the reverse direction, consider a nice optimal Steiner arborescence A for $M(p, [i, j])$. If p has exactly one outgoing arc in A, that arc must end in a point $p' \in P \cap \delta(p) \backslash \{p\}$. Then $A \backslash \{p\}$ is an arborescence rooted at p' that spans $[i, j]$, so it has at least $M(p', [i, j])$ internal vertices. If p has at least two outgoing internal vertices, then let p' be the child of p with the lowest y-coordinate. Since the arborescence is nice, the descendant leaves of p' in A form a subinterval of $[i, j]$ that starts at i. Let q_t be the leaf with the highest y-coordinate among the descendants of p'. If A had strictly fewer internal vertices than $M(p, [i, t]) +$

$M(p, [t+1, j])$, then it would need to include a nice sub-arborescence with fewer internal vertices for at least one of the subproblems $M(p, [i, t])$ or $M(p, [t+1, j])$, but that would contradict the optimality in the definition of the subproblems. □

In the general (two-sided) case, we can have path-like solutions and arborescence-based solutions on both sides, and the two side solutions may or may not share points in L_1. We also need to handle "small" solutions—now these are 2-hop solutions—separately. We now analyze the possible structures of an optimal solution.

Lemma 19 *For any input P inside small strip that has a feasible h-hop broadcast set, there is a minimum h-hop broadcast set D that has one of the following structures:*

- 2-hop*: A solution D that does not not contain any active points from L_2. (Note that such a solution might be optimal even if $h > 2$.)*
- Path-like*: A solution D that consists of two shortest paths, one from s to Q^+ and one from s to Q^-, possibly sharing their first vertex after s.*
- Mixed*: A shortest path on one side, and a nice arborescence on the other side, where the shortest path may share its L_1-vertex with the arborescence.*
- Arborescence-based*: A single arborescence for L_h, which is nice on both sides.*

Proof Suppose that there is no optimal 2-hop solution for P. Thus any optimal solution has active points on $L_{\geqslant 2}$. Let π^+ and π^- be shortest paths to Q^+ and Q^-, respectively. If both π^+ and π^- have at most $h-1$ edges then everything can be reached in h hops. Hence, this is an optimal path-like solution (since it is minimal even for the non-hop-bounded version).

If π^+ has $h+1$ hops and π^- has at most h hops (note that the other case is symmetric), then there is no path-like h-hop broadcast for the right side of the input, that is, for the set $P^* := \{P \cap (\delta(s) \cup \mathcal{S}_{\geqslant 0})\}$. By Lemmas 14 and 17, any minimum-size nice arborescence of P^* gives a minimum h-hop broadcast set for P^*. Either there is a shortest $s \to Q^-$ path whose L_1-vertex is also in some minimum-size arborescence, or there isn't. In both cases, the resulting mixed solution must be optimal. Thus, if exactly one of π^+ and π^- has $h+1$ hops and the other has fewer hops, then there is a mixed optimal solution.

Now suppose both paths have $h+1$ hops. We now now consider an optimal solution D and extend the definition of the *pred* function (as described below) to conclude that D defines a nice arborescence. Let $pred^+$ be the previously defined function in $L_{\leqslant 1} \cup L_i^+$, and let $pred^-$ be the same function for the left side $L_{\leqslant 1} \cup L_i^-$. Note that points in L_1 belong to both sides, but for a point $p \in L_1$ we have $pred^-(p) = pred^+(p) = s$, so this is not an issue. The arborescence defined by this function is nice on both sides by Lemma 17. In addition, since there is no path-like h-hop broadcast set on either side, the active points corresponding to this arborescence form a minimum h-hop broadcast set: by applying Lemma 14 on both sides, we see that the broadcast set corresponding to this arborescence covers all points.

Proof (of Theorem 2) The best 2-hop solution can be found using our 2-hop broadcast algorithm from Theorem 10. The best path-like solution can be found by invoking the narrow-strip broadcast algorithm from Theorem 1, and checking if it satisfies the hop-bound. It remains to describe how to find the best mixed and arborescence-based solutions.

Claim *The best mixed solution can be found in $O(n^5)$ time.*

Proof of Claim Suppose that Q^- can be reached in $t \leqslant h$ hops. Recall from the one-sided case that Q_i^- is the set of points p such that the shortest path from p to Q^- has $i-1$ hops. Thus the set B^- of potential second points of a shortest $s \to Q^-$ path is equal to $B^- := \delta(s) \cap Q_t^-$. (This set can be computed using our algorithm from Theorem 1.) We need to be able to find the potential second points of a nice arborescence. First, we run the one-sided dynamic programming algorithm on the set $P^* := \{P \cap (\delta(s) \cup \mathcal{S}_{\geqslant 0})\}$, which takes $O(n^4)$ time. Let $M(\cdot, [\cdot, \cdot])$ be the resulting dynamic-programming table. We claim that $p \in L_1$ is a potential second point if and only if there is an interval $[i, j]$ such that

$$M(s, [1, m^+]) = M(s, [1, i-1]) + M(p, [i, j]) + M(s, [j+1, m^+]) - 1, \quad (4)$$

where $m^+ = |L_h^+|$.

To prove the claim, first assume that Equality (4) holds. Then the arborescences corresponding to each M-value on the right side are nice minimum arborescences rooted at s, p and s respectively—the fact that s is counted twice explains the -1 term—and so their union together with the edge sp is a minimum arborescence that uses p as as second point. On the other hand, if there is a minimum arborescence using p, then there is a nice one and the set of ancestors of p is an subsequence $q_i, \ldots, q_j$ of L_h^+. The points $q_1, \ldots, q_{i-1}$ and $q_{j+1}, \ldots, q_{m^+}$ are covered by two nice arborescences rooted at s, and the niceness implies that these subtrees only share s. Thus, Equality (4) holds.

Hence, after filling in all entries in the table $M(\cdot, [\cdot, \cdot])$, we can find all potential second points in $O(n^3)$ time by checking all values i, j for each point $p \in L_1$. If there is such a point p in B^-, then the best mixed solution has size $A(s, [1, m^+]) + t - 1$, otherwise it has size $A(s, [1, m^+]) + t$. With standard techniques, an h-hop broadcast set realizing this optimum can be computed within the same time bound. □

Claim *The best arborescence-based solution can be found in $O(n^6)$ time.*

Proof of Claim In order to find the best arborescence-based solution, we modify the one-sided algorithm the following way. For all $p \in L_1 \cup \left(\bigcup_{i=2}^h L_i^+\right)$ we define the subproblems $A^+(p, [i, j])$ as previously, where $[i, j]$ refers to an interval in the last right side level L_h^+. Similarly, we define an ordering on the last left level based on y-coordinates, and define for all $p \in L_1 \cup \left(\bigcup_{i=2}^h L_i^-\right)$ the subproblems $A^-(p, [i, j])$. We can compute these values using the one-sided algorithm on both sides.

It will be convenient to generalize the definitions above as follows. First of all, we extend the definition of $A^+(p, [i, j])$ to include all points $p \in P$—not only the points in $L_1 \cup \left(\bigcup_{i=2}^h L_i^+\right)$—by setting $A^+(p, [i, j]) := \infty$ for $p \in L_{\geqslant 2}^-$. The definition of $A^-(p, [i, j])$ is extended similarly. Finally, we define $A^+(p, [i, j]) := 0$ and $A^-(p, [i, j]) := 0$ for $j = i - 1$.

We also need a third kind of subproblem. Define $A(p, [i, j], [k, \ell])$ as the number of internal vertices in an optimum arborescence rooted at p that has

leaves $q_i^-, \ldots, q_j^-$ in the last left level and from $q_k^-, \ldots, q_\ell^-$ on the last right level. If $p \neq s$, this can be easily expressed:

$$A\big(p, [i, j], [k, \ell]\big) = A^-\big(p, [i, j]\big) + A^+\big(p, [k, \ell]\big) - 1. \tag{5}$$

Note that on the right side of this formula, at least one of the summands is ∞ if $p \in L_{\geqslant 2}$, and possibly for some points in L_1 as well. Since the formula is so simple, we do not need to compute these values explicitly. The only computation for this kind of subproblem is required at the source, for which we require a new notation. Let

$$\begin{aligned}&\mathrm{sep}(i, j, k, \ell)\\ &\quad := \big\{(t, u) \;:\; i - 1 \leqslant t \leqslant j \text{ and } k - 1 \leqslant u \leqslant \ell\big\} \backslash \big\{(i - 1, k - 1), (j, \ell)\big\}.\end{aligned}$$

The set $\mathrm{sep}(i, j, k, l)$ is a shorthand for the set of pairs (t, u) that separate the interval pair $[i, j], [k, l]$ into proper sub-interval-pairs $[i, t], [k, u]$ and $[t + 1, j], [u + 1, l]$. Our formula for the source is the following:

$$\begin{aligned}&A\big(s, [i, j], [k, \ell]\big)\\ &\quad = \min\Bigg(\min_{(t,u)\in \mathrm{sep}(i,j,k,\ell)} \Big(A\big(s, [i, t],\ [k, u]\big) + A\big(s, [t + 1, j], [u + 1, \ell]\big) - 1\Big),\\ &\qquad\qquad \min_{p\in L_1} \Big(A^-\big(p, [i, j]\big) + A^+\big(p, [k, \ell]\big)\Big)\Bigg),\end{aligned}$$

where the first line represents the case when there is an optimal solution where s has at least two active neighbors, while the second line corresponds to the situation where s has only one active neighbor p. The initialization of the values is straightforward:

$$\begin{aligned}A\big(s, [i, i - 1], [k, k - 1]\big) &= 0\\ A\big(s, [i, i], [k, k - 1]\big) &= d_{\mathcal{G}^*}(s, q_i^-)\\ A\big(s, [i, i - 1], [k, k]\big) &= d_{\mathcal{G}^*}(s, q_k^+)\end{aligned}$$

Once the one-sided subproblem values are computed, the above dynamic program can be initialized and computed in increasing order of $(j - i) + (\ell - k)$. The number of subproblems that we need to compute is $O(n^4)$, each of which requires taking the minimum of $O(n^2)$ values. This enables a running time of $O(n^6)$. To prove the correctness of the algorithm, we only need to show that our formulas for L_1 and the source are correct. Again, the inequality $A\big(s, [i, j], [k, \ell]\big) \leqslant \ldots$ is trivial, so we only need to show that there is an optimal solution which has the desired structure.

We start with an optimal arborescence that is nice when restricted to both $L_1 \cup \big(\bigcup_{i=2}^h L_i^-\big)$ and $L_1 \cup \big(\bigcup_{i=2}^h L_i^+\big)$. For a point $p \in L_1$, if the subproblem has a non-empty interval on both sides, then there is a branching at p. The

arborescence can be partitioned into a left and right sub- arborescence, so Eq. (5) holds.

At the source, we only need to explain the case when there is a branching at s, the other case is trivial. Let $p \in L_1$ be the child of s that has the smallest y-coordinate. Since the left and right sub-arborescences are nice, the descendant leaves of p on the left form a starting slice $[i, t]$ of the last level on the left, and the descendant leaves on the right form a starting slice $[k, u]$ of the last level on the right. The rest of the intervals are descendants of the other branches. This demonstrates that the cost of the optimal arborescence can be written as

$$\Big(A\big(s, [i, t], [k, u]\big) + A\big(s, [t + 1, j], [u + 1, \ell]\big)\Big) - 1. \qquad \square$$

The overall algorithm computes the best feasible broadcast set of each type, if it exists: 2-hop, path-like, mixed (for both sides), and arborescence-based Since the minimum broadcast set must have one of these types, the minimum among these is a minimum h-hop broadcast set. The overall running time is $O(n^6)$. $\square$

5 Broadcasting in a Wide Strip

We show that the broadcast problem remains polynomial in a strip of any constant width, or more precisely, it is in XP for the parameter w (the width of the strip). We begin by showing the following key lemma.

Lemma 20 *Let D be the disk centers of a minimum connected dominating set of a unit disk graph on a strip of width w, and let R be an axis parallel rectangle of size $2 \times w$. Then the number of points in $D \cap R$ is at most $\frac{32w}{\sqrt{3}} + 14$.*

Proof Let R' be the set of points inside the strip that are at distance at most 1 from R; thus R' is a $4 \times w$ rectangle. We subdivide R' into cells of diameter 1 by introducing a rectangular grid with side lengths $1/2$ and $\sqrt{3}/2$. Overall, we get $8\lceil \frac{w}{\sqrt{3}/2} \rceil < \frac{16w}{\sqrt{3}} + 8$ cells in R'. Let $\mathcal{G}$ be the unit disk graph spanned by the centers that fall in R'. The points that fall into a grid cell form a clique in $\mathcal{G}$. Let G' be the graph that we get if we contract the vertices of $\mathcal{G}$ in each cell. Let T be a spanning tree of G'. We can represent T in the original graph in the following way. For each edge $uv \in E(T)$ select vertices u', v' of distance at most 1 from the cell of u and v respectively. We know that there are such points since otherwise uv could not be an edge in G'. Since T has at most $(\frac{16w}{\sqrt{3}} + 8) - 1$ vertices, this selection gives us a point set H of size at most $2(\frac{16w}{\sqrt{3}} + 7) = \frac{32w}{\sqrt{3}} + 14$.

Suppose for contradiction that $R \cap D > \frac{32w}{\sqrt{3}} + 14$. We argue that $D' = (D \setminus R) \cup H$ defines a connected dominating set of smaller cost. By our analysis above, we see that the cost is indeed smaller, so we are left to argue that D' is connected and dominating. Notice that $D \cap R$ can only dominate vertices that are inside R', so it is sufficient to argue that all vertices of $\mathcal{G}$ are dominated. This is easy to see because D' has at

least one point in each non-empty cell, and the points in each cell form cliques. It remains to argue that D' is connected. Notice that the set of points in $R' \cap D$ that had a neighbor in D which is outside R' all lie in $R' \setminus R$, so these points are part of D'. So it is sufficient to argue that $V(G) \cap D'$ is connected. This follows from the fact that T is connected and the points of each cell form a clique in $\mathcal{G}$. □

Proof (of Theorem 3) For the sake of simplicity, we start with the one sided case. It is a dynamic programming algorithm that has subproblems for certain $2 \times w$ rectangles, and for each rectangle, all the possible dominating subsets with various connectivity constraints will be considered. More specifically, let $k \in \mathbb{N}$, let $U \subseteq P \cap [k-1, k+1] \times [0, w]$, and let $\sim$ be a binary relation on U. The value of the subproblem $A(k, U, \sim)$ is the minimum size of a set D of active points inside $[0, k+1] \times [0, w]$ for which

- $D \cap [k-1, k+1] \times [0, w] = U$
- D dominates $[0, k] \times [0, w]$
- $u_1 \sim u_2$ if and only if they are connected in the graph spanned by D
- every equivalence class of $\sim$ has a representative in $[k, k+1] \times [0, w]$

By Lemma 20, it is sufficient to consider subproblems where $|U| \leqslant \frac{32w}{\sqrt{3}} + 14$. Let $\mu = \left\lfloor \frac{32w}{\sqrt{3}} + 14 \right\rfloor$. For any value of k, there are at most $\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{\mu} = O(n^{\mu+1})$ such subsets. The relevant values of k are integers between 0 and $2n$. Finally, for any subset U, the number of equivalence relations on U is the number of partitions of U, which is the Bell number $B_{|U|}$. This can be upper bounded by $B_\mu < \mu^\mu = w^{O(w)}$. Thus, the total number of subproblems is $O(n^{\mu+2} w^{O(w)}) = n^{O(w)}$.

For all subsets U of $P \cap [0, 1] \times [0, w]$ with size at most μ, we can compute the equivalence relation $\sim_U$. For all such sets U, we define $A(0, U, \sim_U) = |U|$. For higher values of k, we can compute the subproblems the following way.

When computing $A(k, U, \sim)$ (for which there is a representative of each equivalence class of $\sim$ in $[k, k+1] \times [0, w]$), we first need to find the subproblems $A(k-1, U', \sim')$ for which $U' \cap [k-1, k] \times [0, w] = U \cap [k-1, k] \times [0, w]$. We can only extend this subproblem if $\sim'$ is compatible with $\sim$, i.e., is $s_1, s_2 \in U \cap U'$, then $s_1 \sim' s_2 \Rightarrow s_1 \sim s_2$. We can find these potential subproblems by going through all subproblems $A(k-1, ., .)$, and for each of these, we can decide in polynomial time whether it is compatible with $A(k, U, \sim)$. Overall, computing the solution of a single subproblem takes $n^{O(w)}$ time, so finding the optimal broadcast set in the one sided case can be done in $n^{O(w)}$ time.

For the two sided case, we need to include in the subproblem description the set of active points on both ends. Let $k \in \mathbb{N}$, let $U^- \subseteq P \cap [-k-1, -k+1] \times [0, w]$, $U^+ \subseteq P \cap [k-1, k+1] \times [0, w]$, and let $\sim$ be a relation on $U^- \cup U_+$. Let $B(k, U^-, U^+, \sim)$ be the minimum size of a set D of active points inside $[-k-1, k+1] \times [0, w]$ for which

- $D \cap [-k-1, -k+1] \times [0, w] = U_-$ and $D \cap [k-1, k+1] \times [0, w] = U_+$
- D dominates $[-k, k] \times [0, w]$
- $u_1 \sim u_2$ if and only if they are connected in the graph spanned by D
- every equivalence class of $\sim$ has a representative in $\big([-k-1, -k] \cup [k, k+1]\big) \times [0, w]$.

The number of subproblems is still $n^{O(w)}$, so the running time is also $n^{O(w)}$. □

Surprisingly, as we show in the companion paper, the h-hop version has no algorithm with running time $n^{O(w)}$, unless P = NP.

Remark 2 The goal of our paper is to study the dependency of the complexity of the broadcast problem on the width w of the strip containing the point set P, and so the algorithm presented above is parameterized by w. When k, the size of the solution, is the parameter then one can get $n^{O(\sqrt{k})}$ running time by adapting an algorithm of Marx and Pilipczuk [27]. Their algorithm decides, given a set of n disks and a set Q of n points, if one can cover all the given points by a subset of k of the given disks. This algorithm can be used for DOMINATING SET on a unit disk graph, by letting the set Q coincide with the centers of the disks and scaling each disk by a factor 2. The algorithm of Marx and Pilipczuk is a separator-based divide-and-conquer algorithm. To turn it into an algorithm for CONNECTED DOMINATING SET, we need to track the connectivity through the recursive calls. This can be done by using partitions on the vertices in the "boundary" of the current recursive call, as in [4,12,16]. Doing this does not increase the running time. Finally, we need to turn the algorithm for CONNECTED DOMINATING SET into an algorithm for the broadcast problem. To this end we just need to make sure that when the separator contains the source s, then our guess for part of the solution in the separator includes the source point. This gives us an algorithm with running time $n^{O(\sqrt{k})}$ for the broadcast problem.

6 Concluding Remarks

We have presented algorithms for several variants of the broadcast problem, focusing on the dependency of the running time on the width of the smallest strip containing the points. In a companion paper [11] we complement our results by lower bounds. At the end of that paper we also discuss some directions for further research.

References

1. Alzoubi, K.M., Wan, P., Frieder, O.: Message-optimal connected dominating sets in mobile ad hoc networks. In: Proceedings of the 3rd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2002, 9–11 June 2002, Lausanne, Switzerland, pp. 157–164. ACM (2002). https://doi.org/10.1145/513800.513820
2. Ambühl, C.: An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In: ICALP, Proceedings, pp. 1139–1150 (2005). https://doi.org/10.1007/11523468_92
3. Ambühl, C., Clementi, A.E.F., Ianni, M.D., Lev-Tov, N., Monti, A., Peleg, D., Rossi, G., Silvestri, R.: Efficient algorithms for low-energy bounded-hop broadcast in ad-hoc wireless networks. In: STACS, Proceedings, pp. 418–427 (2004). https://doi.org/10.1007/978-3-540-24749-4_37

4. Bodlaender, H.L., Cygan, M., Kratsch, S., Nederlof, J.: Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. Inf. Comput. **243**, 86–111 (2015)
5. Breu, H.: Algorithmic aspects of constrained unit disk graphs. Ph.D. thesis, University of British Columbia (1996)
6. Cabello, S., Ejčič, M.: Shortest paths in intersection graphs of unit disks. Comput. Geom. **48**(4), 360–367 (2015). https://doi.org/10.1016/j.comgeo.2014.12.003
7. Chazelle, B., Guibas, L.J.: Fractional cascading: I. A data structuring technique. Algorithmica **1**(1), 133–162 (1986)
8. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. Discrete Math. **86**(1–3), 165–177 (1990). https://doi.org/10.1016/0012-365X(90)90358-O
9. Clementi, A.E., Crescenzi, P., Penna, P., Rossi, G., Vocca, P.: A worst-case analysis of an MST-based heuristic to construct energy-efficient broadcast trees in wireless networks. In: STACS, Proceedings, pp. 121–131 (2001)
10. Das, G.K., Das, S., Nandy, S.C.: Range assignment for energy efficient broadcasting in linear radio networks. Theor. Comput. Sci. **352**(1–3), 332–341 (2006). https://doi.org/10.1016/j.tcs.2005.11.046
11. de Berg, M., Bodlaender, H.L., Kisfaludi-Bak, S.: The homogeneous broadcast problem in narrow and wide strips II: lower bounds. Algorithmica (2017). https://doi.org/10.1007/s00453-019-00561-0
12. de Berg, M., Bodlaender, H.L., Kisfaludi-Bak, S., Kolay, S.: An ETH-tight exact algorithm for Euclidean TSP. In: 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, 7–9 October 2018, pp. 450–461. IEEE Computer Society (2018). https://doi.org/10.1109/FOCS.2018.00050. Preprint arxiv:1807.06933
13. de Berg, M., Bodlaender, H.L., Kisfaludi-Bak, S., Marx, D., van der Zanden, T.C.: A framework for ETH-tight algorithms and lower bounds in geometric intersection graphs. In: Proceedings of STOC 2018, pp. 574–586. ACM (2018). https://doi.org/10.1145/3188745.3188854
14. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications, 3rd edn. Springer, Berlin (2008)
15. de Berg, M., Kisfaludi-Bak, S., Woeginger, G.: The complexity of dominating set in geometric intersection graphs. Theor. Comput. Sci. (2018). https://doi.org/10.1016/j.tcs.2018.10.007
16. Deineko, V.G., Klinz, B., Woeginger, G.J.: Exact algorithms for the Hamiltonian cycle problem in planar graphs. Oper. Res. Lett. **34**(3), 269–274 (2006). https://doi.org/10.1016/j.orl.2005.04.013
17. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks **1**(3), 195–207 (1971). https://doi.org/10.1002/net.3230010302
18. Edelsbrunner, H., Guibas, L.J., Stolfi, J.: Optimal point location in a monotone subdivision. SIAM J. Comput. **15**(2), 317–340 (1986)
19. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2006). https://doi.org/10.1007/3-540-29953-X
20. Fuchs, B.: On the hardness of range assignment problems. Networks **52**(4), 183–195 (2008). https://doi.org/10.1002/net.20227
21. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York (1979)
22. Hartenstein, H., Bochow, B., Ebner, A., Lott, M., Radimirsch, M., Vollmer, D.: Position-aware ad hoc wireless networks for inter-vehicle communications: the fleetnet project. In: Proceedings of the 2nd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2001, 4–5 October 2001, Long Beach, CA, USA, pp. 259–262. ACM (2001). https://doi.org/10.1145/501416.501454
23. Hochbaum, D.S., Maass, W.: Approximation schemes for covering and packing problems in image processing and VLSI. J. ACM **32**(1), 130–136 (1985). https://doi.org/10.1145/2455.214106
24. Kuhn, F., Wattenhofer, R., Zhang, Y., Zollinger, A.: Geometric ad-hoc routing: of theory and practice. In: Borowsky, E., Rajsbaum, S. (eds.) Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, 13–16 July 2003, pp. 63–72. ACM (2003). https://doi.org/10.1145/872035.872044
25. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. **11**(2), 329–343 (1982). https://doi.org/10.1137/0211025
26. Marx, D.: Parameterized complexity of independence and domination on geometric graphs. In: Parameterized and Exact Computation, Second International Workshop, IWPEC, Proceedings, pp. 154–165 (2006). https://doi.org/10.1007/11847250_14

27. Marx, D., Pilipczuk, M.: Optimal parameterized algorithms for planar facility location problems using Voronoi diagrams. In: ESA, Proceedings, pp. 865–877 (2015). https://doi.org/10.1007/978-3-662-48350-3_72
28. Masuyama, S., Ibaraki, T., Hasegawa, T.: The computational complexity of the m-center problems on the plane. IEICE Trans. **64**(2), 57–64 (1981)
29. Matsui, T.: Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs. In: JCDCG, Proceedings, pp. 194–200. Springer (1998)
30. van Leeuwen, E.J.: Optimization and approximation on systems of geometric objects. Ph.D. thesis, Universiteit van Amsterdam (2009)
31. Yu, J., Wang, N., Wang, G., Yu, D.: Connected dominating sets in wireless ad hoc and sensor networks—a comprehensive survey. Comput. Commun. **36**(2), 121–134 (2013). https://doi.org/10.1016/j.comcom.2012.10.005
32. Zhang, L., Gao, J.: Load balanced short path routing in wireless networks. In: Proceedings IEEE INFOCOM 2004, The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 1098–1107. IEEE (2004). https://doi.org/10.1109/INFCOM.2004.1356996