The homogeneous broadcast problem in narrow and wide strips

The Homogeneous Broadcast Problem in Narrow and Wide Strips*

Mark de Berg[1], Hans L. Bodlaender[1,2], and Sándor Kisfaludi-Bak[1]

[1] Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
m.t.d.berg@tue.nl; h.l.bodlaender@tue.nl; s.kisfaludi.bak@tue.nl
[2] Department of Computer Science, Utrecht University, The Netherlands

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 almost completely characterize the complexity of both the regular and the hop-bounded versions as a function of the strip width w.

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

* This research was supported by the Netherlands Organization for Scientific Research (NWO) under project no. 024.002.003.

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 $\sum_{p \in P} \mathrm{cost}(\rho(p))$ is minimized. Often the cost of assigning transmission radius x is defined as $\mathrm{cost}(x) = x^{\alpha}$ 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 [9]. In $\mathbb{R}^2$ the problem is NP-hard for any $\alpha > 1$ [7,13], and in $\mathbb{R}^3$ it is even APX-hard [13]. There are also several approximation algorithms [1,7]. For the 2-hop broadcast problem in $\mathbb{R}^2$ an $O(n^7)$ algorithm is known [2] and for any constant h there is a PTAS [2]. 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 $\mathcal{G}_\rho$ in the intersection graph of a set of congruent disks or, in other words, a *unit-disk graph (UDG)*. Because of their relation to wireless networks, UDGs have been studied extensively.

Let $\mathcal{D}$ be a set of congruent disks in the plane, and let $\mathcal{G}_\mathcal{D}$ be the UDG 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 UDG $\mathcal{G}_\mathcal{D}$ with a designated source node s, compute a minimum-size connected dominated set $\Delta \subset \mathcal{D}$ such that $s \in \Delta$.

In the following we denote the dominating set problem by DS, the connected dominating set problem by CDS, and we denote these problems on UDGs by DS-UDG and CDS-UDG, respectively. Given an algorithm for the broadcast problem, one can solve CDS-UDG by running the algorithm n times, once for each possible source point. Consequently, hardness results for CDS-UDG can be transferred to the broadcast problem, and algorithms for the broadcast problem can be transferred to CDS-UDG at the cost of an extra linear factor in the running time. It is well known that DS and CDS are NP-hard, even for planar graphs [14]. DS-UDG and CDS-UDG are also NP-hard [16,19]. The parameterized complexity of DS-UDG has also been investigated: Marx [17] proved that DS-UDG is W[1]-hard when parameterized by the size of the dominating set. (The definition of W[1] and other parameterized complexity classes can be found in the book by Flum and Grohe [12].)

Our contributions. 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. We give an almost complete characterization of the complexity, both for the general and for the hop-bounded version of the problem. More precisely, our results are as follows.

We first study strips of width at most $\sqrt{3}/2$. Unit disk graphs restricted to such *narrow strips* are a subclass of co-comparability graphs [20], for which an $O(nm)$ time CDS algorithm is known [15,4]. (Here m denotes the number of edges in the graph.) The broadcast problem is slightly different because it requires s to be in the dominating set; still, one would expect better running times in this restricted graph class. Indeed, we show that for narrow strips the broadcast problem can be solved in $O(n \log n)$ time. 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. Nevertheless, we show that the h-hop broadcast problem in narrow strips is solvable in polynomial time. Our algorithm runs in $O(n^6)$ and uses a subroutine for 2-hop broadcast, which may be of independent interest: we show that the 2-hop broadcast problem is solvable in $O(n^4)$ time. Our subroutine is based on an algorithm by Ambühl et al. [2] for the non-homogeneous case, which runs in $O(n^7)$ time. This result is deferred to Appendix A.

Second, we investigate what happens for wider strips. We show that the broadcast problem has an $n^{O(w)}$ dynamic-programming algorithm for strips of width w. We prove a matching lower bound of $n^{\Omega(w)}$, conditional on the Exponential Time Hypothesis (ETH). Interestingly, the h-hop broadcast problem has no such algorithm (unless $\mathsf{P} = \mathsf{NP}$): we show this problem is already NP-hard on a strip of width 40. One of the gadgets in this intricate construction can also be used to prove that a CDS-UDG and the broadcast problem are $\mathsf{W}[1]$-hard parameterized by the solution size k. The $\mathsf{W}[1]$-hardness proof is discussed in Section 4. It is a reduction from Grid Tiling based on ideas by Marx [17], and it implies that there is no $f(k)n^{o(\sqrt{k})}$ algorithm for CDS-UDG unless ETH fails.

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 $x(p') > x(p)$. 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 core(p), is $[x(p) - \frac{1}{2}, x(p) + \frac{1}{2}] \times [0, w]$. Note that 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 1. *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 $[x(p) - \frac{1}{2}, x(q) + \frac{1}{2}] \times [0, w]$.

We prove (ii) by contradiction. Suppose that there are $p \in L_{i-1}$ and $q \in L_i$ with $x(p) > x(q) + \frac{1}{2}$. Any shortest path $\pi(s, p)$ must have a point p' inside $[x(q) - \frac{1}{2}, x(q) + \frac{1}{2}] \times [0, w]$, because no edge of the path can jump over this part of the strip. This point p' has level at most $i - 2$ and $q \in \delta(p')$, contradicting that q is at level i.

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. $\square$

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 1(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^+. In most cases it is sufficient to look for broadcast sets with this structure.

Lemma 1. *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.

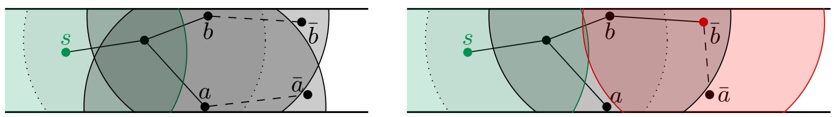

Fig. 1. A swap operation. The edges of the broadcast tree are solid lines.

Proof. We begin by showing that there is a minimum broadcast that intersects both Q^+ and Q^-.

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

Proof of claim. Let D be a minimum broadcast set. Without loss of generality, we may assume that L_2^+ has an active point. It follows that this 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 $\mathcal{S}_{\leqslant 0} \setminus \delta(s)$, since $a \notin \mathrm{core}(s)$ and $\mathrm{core}(s)$ 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 $\bar{a} \in \delta(a) \setminus \delta(b)$ with $x(\bar{a}) > x(b)$ (see Fig. 1 for an example). Since $\delta(b)$ covers $\mathrm{core}(b)$, we have $x(\bar{a}) > x(b) + \frac{1}{2}$, and similarly $x(\bar{b}) > x(a) + \frac{1}{2}$.

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. 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 1, 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.

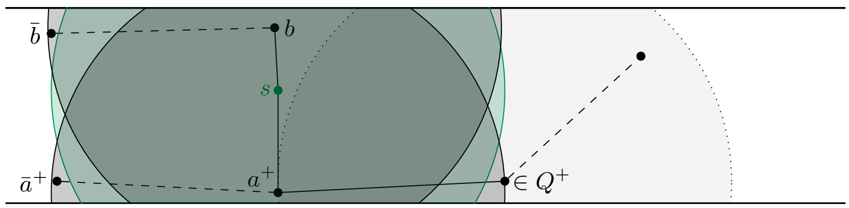

Fig. 2. If $D' \cap L_2^- = \emptyset$, we can still do swaps.

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

Proof of claim. If there is a disk in $D' \cap L_2^-$ as well, 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. 2). Note that $b \in \delta(s)$ since we are in the case $L_2^- = \emptyset$. 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 $\bar{\pi}^+ \cup \pi^-$ $\overline{\pi}^+$ 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 2 below fully characterizes optimal broadcast sets. To deal with the case where Lemma 1 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. 3 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 2. *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 OPT $\geqslant 4$. By Lemma 1 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 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 uniquely covered by $\delta(p_3)$ on the left and right part of the strip. Note that $\delta(a^+)$ covers both $\mathcal{S}_{\geqslant 0} \cap (\delta(p_1) \setminus \delta(s))$ and $\mathcal{S}_{\geqslant 0} \cap (\delta(p_2) \setminus \delta(s))$, as we have seen this happen for regular swap operations in Lemma 1 – similarly, $\delta(a^-)$ covers both $\mathcal{S}_{\leqslant 0} \cap (\delta(p_1) \setminus \delta(s))$ and $\mathcal{S}_{\leqslant 0} \cap (\delta(p_2) \setminus \delta(s))$.

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 1. $\square$

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

Lemma 3. *In $O(n \log n)$ time we can find a bidirectional broadcast if it exists.*

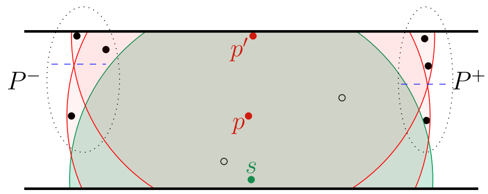

Fig. 3. A bidirectional broadcast.

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 $p, p' \in \mathrm{core}(s)$ such that $\delta(p)$ covers $P^-_{\leqslant i} \cup P^+_{\leqslant j}$ and $\delta(p')$ covers $P^-_{>i} \cup P^+_{>j}$; see Fig. 3.

Now for a point $p \in \mathrm{core}(s)$, 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 $p, p' \in \mathrm{core}(s)$ such that $Z^-_{\leqslant}(p) \geqslant Z^-_{>}(p')$ and $Z^+_{\leqslant}(p) \geqslant Z^+_{>}(p')$.

It is easy 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 $p \in \delta(s)$. 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 ν in $\mathcal{T}$, let $F(\nu) := \{\delta(u_i) : u_i \text{ is stored in the subtree rooted at } \nu\}$. We start by computing at each node ν the intersection of the disks in $F(\nu)$. More precisely, for each ν we compute the region $I(\nu) := \mathrm{core}(s) \cap \bigcap F(\nu)$. Notice that $I(\nu)$ is y-monotone and convex, and each disk $\delta(u_i)$ contributes at most one arc to $\partial I(\nu)$. (Here $\partial I(\nu)$ refers to the boundary of $I(\nu)$ that falls inside $\mathcal{S}$.) Moreover, $I(\nu) = I(\text{left-child}(\nu)) \cap I(\text{right-child}(\nu))$. Hence, we can compute the regions $I(\nu)$ of all nodes ν 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 $p \in \mathrm{core}(s)$ by searching in $\mathcal{T}$, as follows. Suppose we arrive at a node ν. If $p \in I(\text{left-child}(\nu))$, then descend to right-child(ν), otherwise descend to left-child(ν). 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 $I(\nu)$ is a convex region, we can check if $p \in I(\nu)$ in $O(1)$ time if we can locate the position of p_y in the sorted list of y-coordinates of the vertices of $\partial I(\nu)$.

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 [6]. 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. $\square$

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 UDGs by Cabello and Jejčič [5].

Lemma 4. *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 [3,11]. To compute $Q \cap \left(\bigcup_{p\in P} \delta(p)\right)$ we proceed similarly, except that we use the farthest-point Voronoi diagram [3]. $\square$

Lemma 5. *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 4. $\square$

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

Proof. 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 5. 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 4, and set $P' := P' \setminus Q_i^+$.
4. If $Q_i^+ = \emptyset$, return failure.
5. Compute Q^- using Lemma 5. 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 4, 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 4. Finally, we need to check whether there is a feasible minimum broadcast with the bidirectional structure. Lemma 3 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^+: 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 4. 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. 4). 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. □

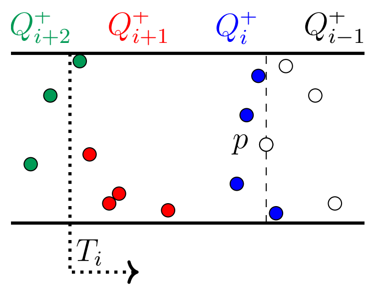

Fig. 4. The levels Q^+_i computed by the algorithm.

Remark 1. If we apply this algorithm to every disk as source, we get an $O(n^2 \log n)$ algorithm for CDS in narrow strip UDGs. We can compare this to $O(mn)$, the running time that we get by applying the algorithm for co-comparability graphs [4]. Note that in the most difficult case, when the size of the minimum connected dominating set is at most 3, the unit disk graph has constant diameter, which implies that the graph is dense, i.e., the number of edges is $m = \Omega(n^2)$. Hence, we get an (almost) linear speedup for the worst-case running time.

3 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 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 $\mathcal{G}$, and we define L^+_i and L^-_i to be 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 2. (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.

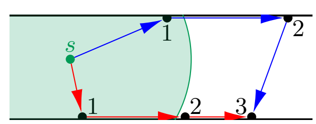

Fig. 5. Two different arborescences, with vertices labeled with their level. The red arborescence does not define a feasible broadcast for $h = 3$, since it would take four hops to reach the top right node.

Remark 2. 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 [4], 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 6 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 activating all non-leaf nodes of the arborescence. We denote the broadcast set obtained from an arborescence A by D_A.

Lemma 6. *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 1(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 1(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 1(i) shows that the disks of a shortest $s \to p$ path form a feasible broadcast set, which is a path-like solution. $\square$

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. 5. Lemma 6 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 [10], and obtain an optimal solution from this tree.[3] 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: $y(u') < y(v') \Rightarrow y(u) < y(v)$. Intuitively, a nice arborescence is one consisting of paths that can be ordered vertically in a consistent manner, see the left of Fig. 6. 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 3. *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 1(i), all points to the left of p are covered by the

[3] 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.

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 $v \in L_{i+1} \cap (\delta(q) \setminus \delta(p))$. Since $\delta(q) \setminus \delta(p)$ 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 7. *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 $\pi_p \cup (D \cap L_{h-1}) \subseteq D$. □

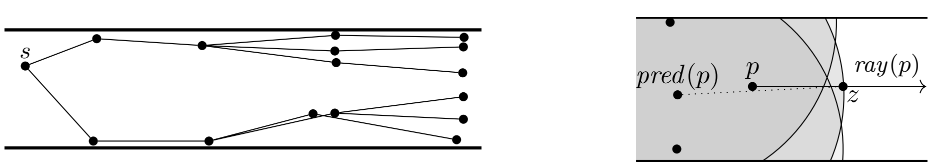

Fig. 6. Left: A nice Steiner arborescence. Note that arc crossings are possible. Right: Defining the *pred* function.

Lemma 8. *Every optimal broadcast set D has a nice compatible arborescence.*

Proof sketch. To find a nice compatible arborescence we will associate a unique arborescence with D. To this end, we define for each $p \in (D \cup L_h) \setminus \{s\}$ 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 3 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 $pred : (D \cup L_h) \setminus \{s\} \to D$ the following way. Consider a point $p \in (D \cup L_h) \setminus \{s\}$ and let i be its level. Let $ray(q)$ be the horizontal ray emanating from q to the right; see the right of Fig. 6. It follows from Observation 1(iii) that $ray(q)$ cannot enter any disk from level $i-1$. We can prove that any point $p \in D \cap L_h$ is contained in a disk from p's previous level, so $pred(p)$ 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 $\bigcup\{\delta(p) : p \in L_i \cap D\}$. □

Proof. Recall that $pred(p)$, for $p \in L_i \cap D$, is the center of the level $i-1$ disk which has $z = ray(p) \cap \partial_{i-1}$ on its boundary. 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.

Let A be the directed graph defined by the edges $pred(p)p$ for each $p \in (D \cup L_h) \setminus \{s\}$. 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 4. *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 6. By Lemma 8, 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:

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

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 Equation (1) 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 $p' \in P \cap \delta(p) \setminus \{p\}$. Then $A \setminus \{p\}$ 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 sub- interval 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 less internal vertices than $M(p, [i, t]) + M(p, [t+1, j])$, then it would need to include a nice sub-arborescence with less 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.

Theorem 5. *The h-hop broadcast problem inside a strip of width at most $\sqrt{3}/2$ can be solved in $O(n^6)$ time.*

Proof. We first analyze the possible structures of an optimal solution.

Claim. 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 of claim. 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, then there is no path-like h-hop broadcast for the right side of the input, that is, for the set $P^* := \{P \cap (\delta(s) \cup \mathcal{S}_{\geqslant 0})\}$. Let T be a minimum-size nice arborescence for P^*. By Lemma 6 and Lemma 8, T gives a minimum h-broadcast set for P^*. Either there is a shortest $s \to Q^-$ path whose L_1-vertex is also in a 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 $\mathit{pred}^-(p) = \mathit{pred}^+(p) = s$, so this is not an issue. The arborescence defined by this function is nice on both sides by Lemma 8. 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 6 on both sides, we see that the broadcast set corresponding to this arborescence covers all points. □

The best 2-hop solution can be found using our planar 2-hop broadcast algorithm from Theorem 11. The best path-like solution can be found by invoking the narrow-strip broadcast algorithm from Theorem 2, 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 2.) 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

$$M(s, [1, m^+]) = M(s, [1, i-1]) + M(p, [i, j]) + M(s, [j+1, m^+]) - 1, \quad (2)$$

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

To prove the claim, first assume that Equality (2) 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 $q_i, \dots, q_j$ of L_h^+. The points $q_1, \dots, q_{i-1}$ and $q_{j+1}, \dots, q_{m^+}$ are covered by two nice arborescences rooted at s, and the niceness implies that these subtrees only share s. Thus, Equality (2) 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. $\square$

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 $p \in L_1 \cup \big(\bigcup_{i=2}^{h} L_i^+\big)$ 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 $p \in L_1 \cup \big(\bigcup_{i=2}^{h} L_i^-\big)$ 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 $L_1 \cup \big(\bigcup_{i=2}^{h} L_i^+\big)$—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 $q_i^-,\dots,q_j^-$ in the last left level and from $q_k^-,\dots,q_\ell^-$ on the last right level. If $p \neq s$, this can be easily expressed:

$$A\big(p,[i,j],[k,\ell]\big) = A^-\big(p,[i,j]\big) + A^+\big(p,[k,\ell]\big) - 1. \tag{3}$$

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

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

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:

$$\begin{aligned}&A\big(s,[i,j],[k,\ell]\big) = \\ &\min \begin{cases} \displaystyle\min_{(t,u)\in \mathrm{sep}(i,j,k,\ell)} \Big(A\big(s,[i,t],\ [k,u]\big) + A\big(s,[t+1,j],[u+1,\ell]\big) - 1\Big) \\ \hfill \text{if branching at } s, \\ \displaystyle\min_{p\in L_1} \Big(A^-\big(p,[i,j]\big) + A^+\big(p,[k,\ell]\big)\Big) \\ \hfill \text{otherwise.} \end{cases}\end{aligned}$$

The initialization of the values is straightforward:

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

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 require 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 $A\big(s,[i,j],[k,\ell]\big) \leqslant \dots$ 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 equation (3) 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 $O(n^6)$. $\square$

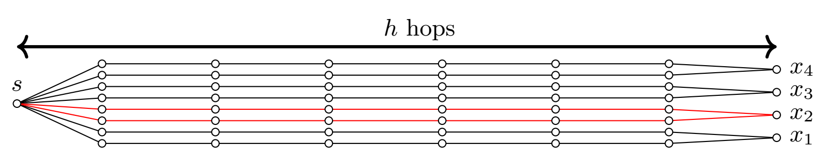

Fig. 7. The gadget representing the variables. The red paths form the x_2-string.

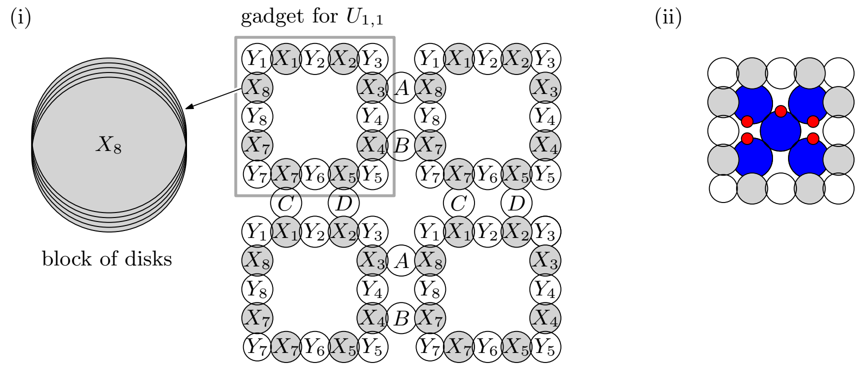

Fig. 8. (i) The construction by Marx. (ii) The idea behind our construction.

4 A parameterized look at CDS-UDG

In this section we prove that CDS-UDG is W[1]-hard parameterized by the solution size; our proof heavily relies on the proof of the W[1]-hardness of DS-UDG by Marx [17].

The construction by Marx for DS-UDG. Marx uses a reduction from Grid Tiling [8] (although he does not explicitly state it this way). In a grid-tiling problem we are given an integer k, an integer n, and a collection $\mathcal{S}$ of k^2 non-empty sets $U_{a,b} \subseteq [n] \times [n]$ for $1 \leqslant a, b \leqslant k$. The goal is to select an element $u_{a,b} \in U_{a,b}$ for each $1 \leqslant a, b \leqslant k$ such that

– If $u_{a,b} = (x, y)$ and $u_{a+1,b} = (x', y')$, then $x = x'$.
– If $u_{a,b} = (x, y)$ and $u_{a,b+1} = (x', x')$, then $y = y'$.

One can picture these sets in a $k \times k$ matrix: in each cell (a, b), we need to select a representative from the set $U_{a,b}$ so that the representatives selected from horizontally neighboring cells agree in the first coordinate, and representatives from vertically neighboring sets agree in the second coordinate.

Marx's reduction places k^2 gadgets, one for each $U_{a,b}$. A gadget contains 16 blocks of disks, labeled $X_1, Y_1, X_2, Y_2, \ldots, X_8, Y_8$, that are arranged along the edges of a square—see Fig. 8(i). Initially, each block X_ℓ contains n^2 disks, denoted by $X_\ell(1), \ldots, X_\ell(n^2)$ and each block Y_ℓ contains $n^2 + 1$ disks denoted by $Y_\ell(0), \ldots, Y_\ell(n^2)$. The argument j of $X_\ell(j)$ can be thought of as a pair (x, y) with $1 \leqslant x, y \leqslant n$ for which $f(x, y) := (x - 1)n + y = j$. Let $f^{-1}(j) = (\iota_1(j), \iota_2(j)) = (1 + \lfloor j/n \rfloor, 1 + (j \mod n))$. For the final construction, in each gadget at position (a, b), delete all disks $X_\ell(j)$ for each $\ell = 1, \ldots, 8$ and $(\iota_1(j), \iota_2(j)) \notin U_{a,b}$. This deletion ensures that the gadgets represent the corresponding set $U_{a,b}$. The construction is such that a minimum dominating set uses only disks in the X-blocks, and that for each gadget (a, b) the same disk $X_\ell(j)$ is chosen for each $1 \leqslant \ell \leqslant 8$. This choice signifies a specific choice $u_{a,b} = (x, y)$. To ensure that the choice for $u_{a,b}$ in the same row and column agrees on their first and second coordinate, respectively, there are special connector blocks between neighboring gadgets. The connector blocks are denoted by A, B, C and D in Fig. 8(i), and they each contain $n + 1$ disks—see Section 4 for further details.

Our construction for CDS-UDG. To extend the construction to CDS-UDG, we have to make sure there is a minimum-size dominating set that is connected. This requires two things. First, we must add new disks inside the gadgets—that is, in the empty space surrounded by the X- and Y-blocks—to guarantee a connection between all chosen $X_\ell(j)$ disks without interfering with the disks in the Y-blocks. Second, we need to connect all the different gadgets. This time, in addition to avoiding the Y-blocks, we also need to avoid interfering with the connector blocks.

The idea is as follows. Inside each gadget we add several pairs of disks, consisting of a *parent* disk and a *leaf* disk. The parent disks are placed such that, for any choice of one disk from each of the X-blocks, the parent disks together with the eight chosen disks from the X-blocks form a connected set. Moreover, the parent disks do not intersect any disk in a Y-block. See Fig. 8(ii) for an illustration; the parent disks are blue in the figure. For each parent disk we add a leaf disk —the red disks in the figure—that only intersects its parent disk. This ensures there is a minimum dominating set containing all the parent disks, which in turn implies that any minimum dominating set for the gadget is connected.

In Fig.8(ii) we used disks of different sizes. Unfortunately this is not allowed, which makes the construction significantly more tricky. To be able to place the pairs in a suitable way, we need to create more space inside the gadget. To this end we use a gadget consisting of 16 (instead of eight) X- and Y-blocks. This will also give us sufficient space to put parent-leaf pairs in between the gadgets, so the dominating sets from adjacent gadgets are connected through the parent disks; see Section 4 for details. Thus the size of a minimum connected dominating set in the new construction is equal to the size of a minimum dominating set in the old construction plus the number of parent disks. Hence, we can decide if the GRID TILING instance has a solution by checking the size of the minimum connected dominating set in our construction. Thus CDS-UDG is W[1]-hard. Moreover, if the Exponential Time Hypothesis (ETH) holds, then there is no algorithm for GRID TILING that runs in time $f(k)n^{o(k)}$ [8]. We thus obtain the following result.

Theorem 6. *The broadcast problem and* CDS-UDG *are* W[1]*-hard when parameterized by the solution size. Moreover, there is no $f(k)n^{o(\sqrt{k})}$ algorithm for these problems, where n is the number of input disks and k is the size of the solution, unless ETH fails.*

Remark. Using a modified version of an algorithm by Marx and Pilipczuk [18], it is possible to construct an algorithm for CDS-UDG with running time $n^{O(\sqrt{k})}$.

Some details of the construction in [17]. In every block, the place of each disk center is defined with regard to the midpoint of the block, $(x(z), y(z))$. The center of each circle is of the form $(x(z)+\alpha\epsilon, y(z)+\beta\epsilon)$ where $x(z), y(z), \alpha$ and β are integers, and $\epsilon > 0$ a small constant. We say that the *offset* of the disk centered at $(x(z)+\alpha\epsilon, y(z)+\beta\epsilon)$ is (α, β). Note that $|\alpha|, |\beta| \leqslant n$, and $\epsilon < n^{-2}$, so the disks in a block all intersect each other. The offsets of X and Y-blocks are defined as follows.

$$\begin{array}{ll}
\text{offset}(X_1(j)) = (j, -\iota_2(j)) & \text{offset}(Y_1(j)) = (j+0.5, j+0.5) \\
\text{offset}(X_2(j)) = (j, \iota_2(j)) & \text{offset}(Y_2(j)) = (j+0.5, -n) \\
\text{offset}(X_3(j)) = (-\iota_1(j), -j) & \text{offset}(Y_3(j)) = (j+0.5, -j-0.5) \\
\text{offset}(X_4(j)) = (\iota_1(j), -j) & \text{offset}(Y_4(j)) = (-n, -j-0.5) \\
\text{offset}(X_5(j)) = (-j, \iota_2(j)) & \text{offset}(Y_5(j)) = (-j-0.5, -j-0.5) \\
\text{offset}(X_6(j)) = (-j, -\iota_2(j)) & \text{offset}(Y_6(j)) = (-j-0.5, n) \\
\text{offset}(X_7(j)) = (\iota_1(j), j) & \text{offset}(Y_7(j)) = (-j-0.5, j+0.5) \\
\text{offset}(X_8(j)) = (-\iota_1(j), j) & \text{offset}(Y_8(j)) = (n, j+0.5)
\end{array}$$

We remark some important properties. First, two disks can intersect only if they are in the same or in neighboring blocks. Consequently, one needs at least eight disks to dominate a gadget. The second important property is that disk $X_\ell(j)$ dominates exactly $Y_\ell(j), \ldots, Y_\ell(n^2)$ from the "previous" block Y_ℓ, and $Y_{\ell+1}(0), \ldots, Y_{\ell+1}(j-1)$ from the "next" block $Y_{\ell+1}$. This property can be used to prove the following key lemma.

Lemma 9 (Lemma 1 of [17]). *Assume that a gadget is part of an instance such that none of the blocks Y_i are intersected by disks outside the gadget. If there is a dominating set Δ of the instance that contains exactly $8k^2$ disks, then there is a* canonical *dominating set Δ' with $|\Delta'| = |\Delta|$, such that for each gadget $\mathcal{G}$, there is an integer $1 \leqslant j^G \leqslant n$ such that Δ' contains exactly the disks $X_1(j^G), \ldots, X_8(j^G)$ from $\mathcal{G}$.*

In the gadget $G_{a,b}$, the value j defined in the above lemma represents the choice of $s_{a,b} = (\iota_1(j), \iota_2(j))$ in the grid tiling problem. Our deletion of certain disks in X-blocks ensures that $(\iota_1(j), \iota_2(j)) \in U_{a,b}$. Finally, in order to get a feasible grid tiling, gadgets in the same row must agree on the first coordinate, and gadgets in the same column must agree on the second coordinate. These blocks have $n+1$ disks each, with indices $0, 1, \ldots, n$. We define the offsets in the connector gadgets the following way.

$$\begin{array}{ll}
\text{offset}(A_j) = (-j-0.5, -n^2-1) & \text{offset}(B_j) = (j+0.5, n^2+1) \\
\text{offset}(C_j) = (n^2+1, -\iota_2(j)) & \text{offset}(D_j) = (-n^2-1, \iota_2(j))
\end{array}$$

Using this definition, it is easy to prove the following lemma.

Lemma 10. *Let Δ be a canonical dominating set. For horizontally neighboring gadgets $\mathcal{G}$ and H representing j_G and j_H, the disks of the connector block A are dominated if and only if $\iota_1(j_G) \leqslant \iota_1(j_H)$; the disks of B are dominated if and only if $\iota_1(j_G) \geqslant \iota_1(j_H)$. Similarly, for vertically neighboring blocks G' and H', the disks of block C are dominated if and only if $\iota_2(j_{G'}) \leqslant \iota_2(j_{H'})$; the disks of D are dominated if and only if $\iota_2(j_{G'}) \geqslant \iota_2(j_{H'})$.*

With the above lemmas, the correctness of the reduction follows. A feasible grid tiling defines a dominating set of size $8k^2$: in gadget $G_{a,b}$, the dominating disks are $X_\ell\left(f(s_{a,b})\right),\ \ell = 1, \ldots, 8$. On the other hand, if there is a dominating set of size $8k^2$, then there is a canonical dominating set of the same size that defines a feasible grid tiling.

Details of the CDS-UDG construction. To extend the construction to CDS-UDG, we want to make sure that minimum-size dominating set is connected. This requires two things. First, we must add new disks "inside" the gadgets — that is, in the empty space surrounded by the X and Y-blocks — such that a canonical minimum dominating set includes some new disks that connect the chosen $X_\ell(j)$ disks without interfering with disks in the Y-blocks. Second, we need to connect all the different gadgets. This time in addition to avoiding the Y-blocks, we also need to avoid interfering with the connector blocks.

In order to have enough space, our gadgets contain 16 X-blocks and 16 Y-blocks instead of eight. The offsets of disks inside the blocks are not modified: we use the same building blocks. Fig. 10 shows how we arrange these blocks, and depicts the connector block placement.

The analogue of Lemma 9 and Lemma 10 are true here; we have a construction that could be used to prove the W[1]-hardness of DS-UDG, with canonical sets of size $16k^2$, that contain one disk from each X-block and X'-block. We extend this construction with parent-leaf pairs so that we have canonical dominating sets that span a connected subgraph.

The most important property of the blocks that we use is that for a small enough value ϵ, the boundaries of the disks in a block all lie inside a small width annulus - for this reason, the blocks in our pictures are depicted with thick boundary disks. In order for a parent disk p to intersect every disk in a block it is sufficient if the boundary of p crosses this annulus.

We are going to add 72 extra disks to every gadget, and 4 "connector" disks between every pair of horizontally or vertically neighboring gadgets, resulting in canonical dominating sets of size $16k^2 + 36k^2 + 4k(k-1) = 56k^2 - 4k$ (Note that only the parent disks are included in the canonical set). In other words, the new construction has a connected dominating set of size $56k^2 - 4k$ if and only if there is a feasible grid tiling.

Inside any of the blocks, all offsets are in the rectangle with bottom left $(-n^2-1, -n^2-1)$ and top right (n^2+1, n^2+1). Consequently, every circle in the block with center (r,s) passes through the square with bottom left $\big((-n^2-1)\epsilon, 1-(n^2+1)\epsilon\big)$ and top right $\big((n^2+1)\epsilon, 1+(n^2+1)\epsilon\big)$. There are three similar squares that also have this property, which we can get by rotating the square around the midpoint of the block by 90, 180 and 270 degrees. Consequently, a unit disk that contains such a square intersects all the disks in the given block. For an example with $n = 3$ and $\epsilon = 0.02$ for the block X_2, see Fig. 9.

Connecting neighboring gadgets. For a pair of horizontally neighboring gadgets, we add two pairs of disks that connect X_3' from the left gadget to X_8' in the right gadget. This arrangement is depicted on the left of Fig. 12. The parent disk with

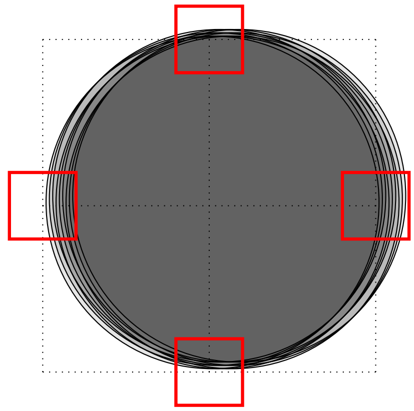

Fig. 9. Circles in a block. The squares intersect every disk in the block.

center T_1 intersects every disk in the block X'_3 of the left gadget, and the other parent intersects every disk in the block X'_8. The two leaf disks (red disks in the figure) only intersect their parent. Let the origin be the center of the block X'_3 in the left gadget. The coordinates for the disk centers are:

$$T_1 = (1.3, 0.4) \quad U_1 = (2, 1.55)$$
$$T_2 = (2.7, -0.4) \quad U_2 = (2, -1.55)$$

We use a rotated version of these four disks for vertical connections, where the parents connect X'_5 from the upper gadget and X'_2 from the lower gadget.
Disks inside gadgets. We begin by adding eight disk pairs to the center. The parents are arranged in a square, touching the neighbors, and the leafs are placed so that it is possible to connect from the outside on each side. See Fig. 11 for a picture: the corresponding leaf disks have parallel lines as a pattern.

Let $\delta > 0$ be a small constant to be specified later. From now on, we fix the origin in the center of the bottom left block, Y_7. The coordinates of the disks centers are given below; in each pair we specify the coordinates of a parent and its leaf.

$$(6,6),(6-\delta,6) \quad (8,6),(8,6+4\delta) \quad (10,6),(10,6-\delta) \quad (10,8),(10-4\delta,8)$$
$$(10,10),(10,10+\delta) \quad (8,10),(8,10-4\delta) \quad (6,10),(6,10+\delta) \quad (6,8),(6+4\delta,8)$$

In order to connect the X-blocks, we need to connect the blocks of each side to the central disks. For this purpose, we are going to use a zigzag pattern of disks. The first parent disk intersects all disks in X_6 and X_7 (i.e., it crosses the small squares of X_6 and X_7 that are facing the inside of the gadget). The second

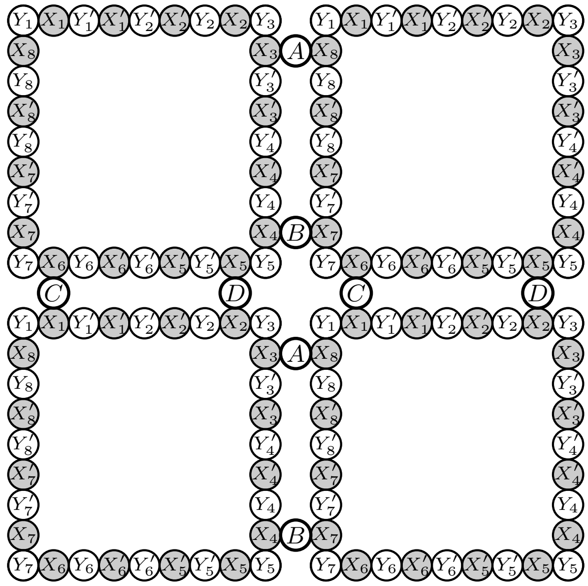

Fig. 10. Connecting neighboring gadgets

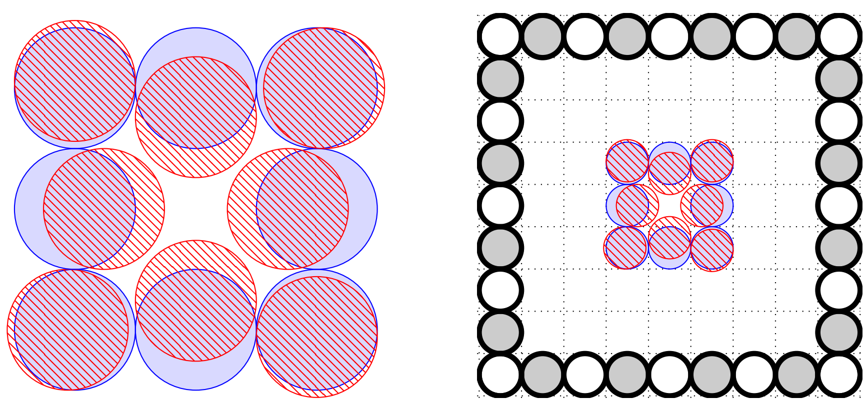

Fig. 11. Left: the circles in the center of every gadget; Right: placement inside a gadget

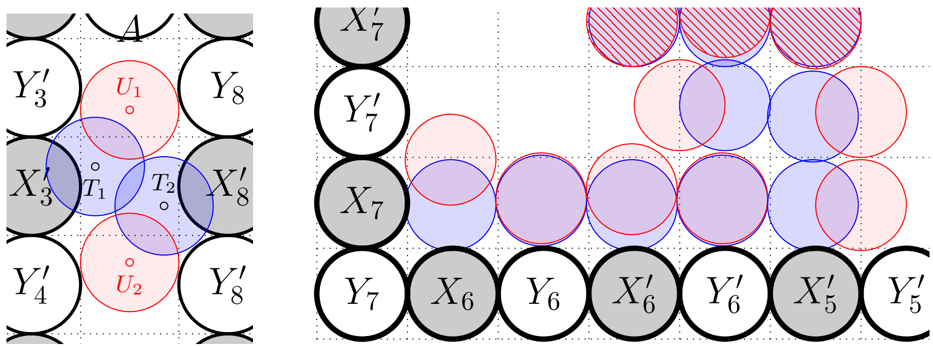

Fig. 12. Left: connecting horizontally; right: connecting one side to the middle.

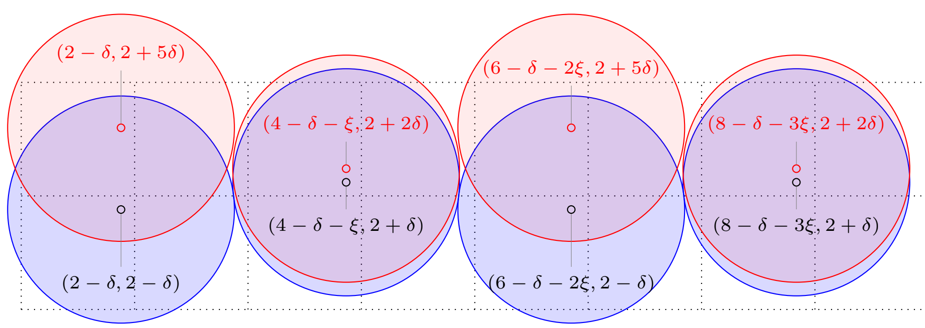

Fig. 13. The zig-zag arrangement

parent is above the block Y_6, but it is disjoint from it. The next with center p_3 intersects all disks in X_6' , and the disk around p_4 is disjoint from the disks in Y_6'. Finally, the disk around p_5 intersects all disks in X_5'. See Fig. 13 for an example. The leafs follow a more complicated pattern. In our zigzag pattern, two neighboring parents touch each other. We need them to have distance 2δ along the y axis, so the distance along the x axis is $\sqrt{4-4\delta^2}$. Let $\xi = 2 - \sqrt{4-4\delta^2}$. Note that

$$2 - \delta^2 - \delta^4 < \sqrt{4-4\delta^2} < 2 - \delta^2,$$

so $\delta^2 < \xi < \delta^2 + \delta^4$. We add two more disk pairs to this pattern, and some modifications to the leafs. These seven disk pairs are depicted on the right side of Fig. 12. We list the coordinates of the disk centers below.

$$\begin{aligned}
p_1 &= (2-\delta, 2-\delta) & \ell_1 &= (2-\delta, 3-\delta)\\
p_2 &= (4-\delta-\xi, 2+\delta) & \ell_2 &= (4-\delta-\xi, 2+2\delta)\\
p_3 &= (6-\delta-2\xi, 2-\delta) & \ell_3 &= (6-\delta-2\xi, 1+5\delta)\\
p_4 &= (8-\delta-3\xi, 2+\delta) & \ell_4 &= (8-\delta-3\xi, 2+2\delta)\\
p_5 &= (10-\delta-4\xi, 2-\delta) & \ell_5 &= (11, 2-\delta)\\
p_6 &= (10-\delta-4\xi, 4-\delta) & \ell_6 &= (11, 4)\\
p_7 &= (8, 4+3\delta) & \ell_7 &= (7, 4+3\delta)
\end{aligned}$$

Our final gadget can be attained by rotating the above seven disk pairs around the center $(8,8)$ by $90, 180$ and 270 degrees: see Fig. 14. We added the spanned edges of a canonical dominating set to this picture.

We can now turn to the proof of the following theorem.

Theorem 7. *The* CDS-UDG *problem is* W[1]*-hard.*

Proof. A feasible grid tiling defines $16k^2$ disks: in gadget (a,b), we include the disks $X_\ell\left(f(s_{a,b})\right)$ and $X'_\ell\left(f(s_{a,b})\right)$ for all $\ell = 1,\dots,8$. We add all parent disks of the construction, this results in a connected dominating set of size $56k^2-4k$. In the other direction, if there is a connected dominating set of size $56k^2-4k$, then there is a canonical dominating set of the same size, whose disks inside X-blocks and X'-blocks define a feasible grid tiling. Thus, it is sufficient to prove that the intersection patterns are as described.

It can be verified using the coordinates that our final leaf disks only intersect their parent disk, and also that the parent disks form a connected subgraph both inside gadgets and at every connection. We need to show that the parents inside a gadget connect all the X-blocks of a gadget, and that the horizontal and vertical connectors intersect the two X-blocks that they need to connect. In all of these cases, it is sufficient to show that the parent disk contains one of the four squares that we associated with each block. For connector disks, it is easy to see that the center of one of the four squares is covered by the interior of the corresponding parent disk (i.e., the square around $(1,0)$ is contained in the

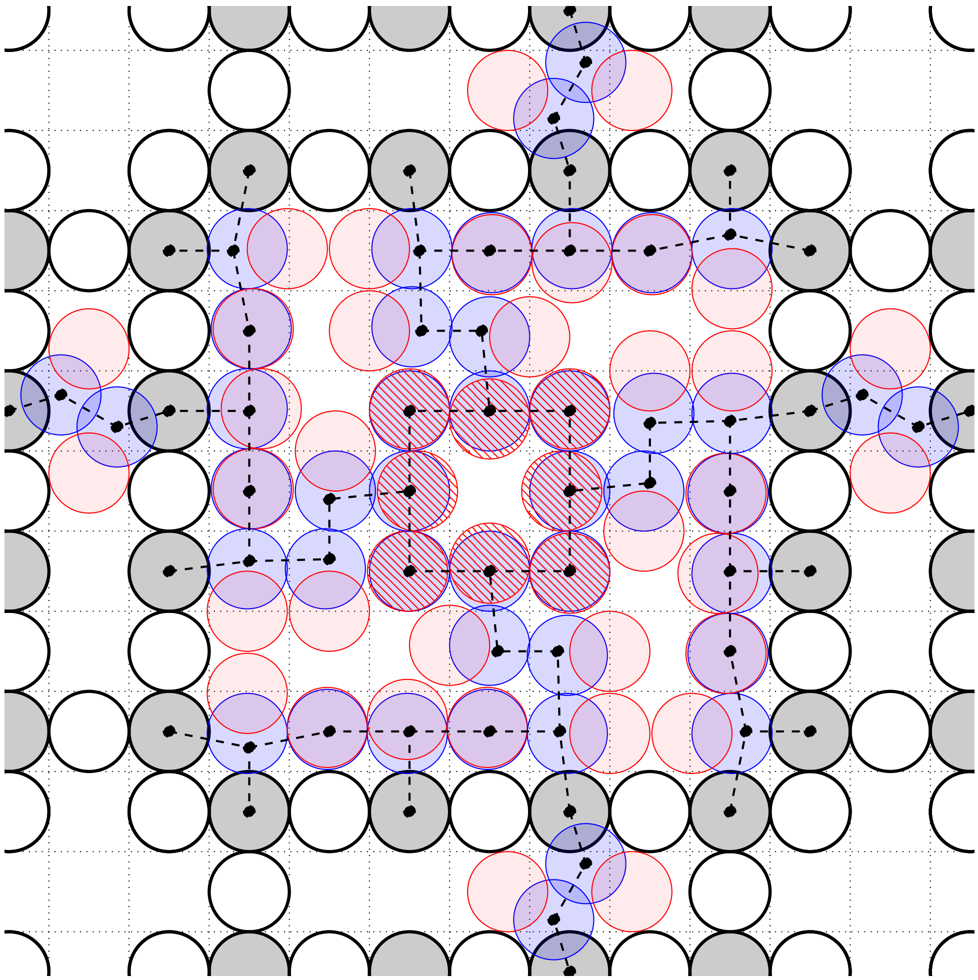

Fig. 14. A gadget in the final construction. The dashed lines are spanned edges of a canonical dominating set.

interior of $\delta(T_1)$). By choosing a small enough value for ϵ, the square is contained in the parent disk.

For the inner connections of gadgets, it is sufficient to show that the inner squares of X_7, X_6, X_6' and X_5' are contained in $\delta(p_1), \delta(p_1), \delta(p_3)$ and $\delta(p_5)$ respectively: the other sides have the same containments since the rotation around $(8,8)$ by $90, 180$ and 270 degrees are automorphisms on the small squares. The largest distance between parent disk and the corresponding small square is at $\delta(p_5)$ and the inner small square of block X_5'. The farthest corner of the square from p_5 is $\big(10 + (n^2+1)\epsilon, 1 - (n^2+1)\epsilon\big)$. Let $\epsilon < \frac{1}{2n^3}$ and $\delta < 1$. The distance squared from p_5 has to be at most 1:

$$\begin{aligned}
&\big(10 - \delta - 4\xi - (10 + (n^2+1)\epsilon)\big)^2 + \big(2 - \delta - (1 - (n^2+1)\epsilon)\big)^2 \\
<&\left(\delta + 4\delta^2 + \delta^4 + \frac{1}{n}\right)^2 + \left(1 - \delta + \frac{1}{n}\right)^2 \\
=&1 - 2\delta + \frac{4}{n} + O\left(\frac{\delta}{n}\right) + O(\delta^2)
\end{aligned}$$

Let $\delta = \frac{1}{\sqrt{n}}$. For n large enough,

$$1 - 2\delta + \frac{4}{n} + O\left(\frac{\delta}{n}\right) + O(\delta^2) \;=\; 1 - \frac{2}{\sqrt{n}} + \frac{4}{n} + O\left(\frac{1}{n\sqrt{n}}\right) + O\left(\frac{1}{n}\right) < 1.$$

Note that the coordinates of each point can be represented with $O(\log n)$ bits, since a precision of c/n^4 is sufficient for the construction. □

We can let one of the blue parent disks be the source disk: in this way, the minimum broadcast sets equal the minimum connected dominating sets. We get the following corollary.

Corollary 1. *The broadcast problem is* W[1]*-hard parameterized by the size of the broadcast set.*

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).

Theorem 8. *The broadcast problem and* CDS-UDG *can be solved in $n^{O(w)}$ time on a strip of width w. Moreover, there is no algorithm for* CDS-UDG *or the broadcast problem with runtime $f(w)n^{o(w)}$ unless ETH fails.*

We begin by showing the following key lemma.

Lemma 11. *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 1-neighborhood of R inside the strip (so 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 $\mathcal{G}$. $\square$

Proof (Proof of Theorem 8). 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 11, it is sufficient to consider subproblems where $|U| \leqslant \frac{32w}{\sqrt{3}} + 14$. Let $\mu = \lfloor \frac{32w}{\sqrt{3}} + 14 \rfloor$. 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, the h-hop version has no $n^{O(w)}$ algorithm (unless $\mathsf{P} = \mathsf{NP}$).

6 The hardness of *h*-hop broadcast in wide strips

The goal of this section is to prove the following theorem.

Theorem 9. *The h-hop broadcast problem is* NP*-complete in strips of width* 40*.*

(The theorem of course refers to the decision version of the problem: given a point set P, a hop bound h, and a value K, does P admit an h-hop broadcast set of size at most K?) Our reduction is from 3-SAT. Let $x_1, x_2, \ldots x_n$ be the variables and $C_1, \ldots, C_m$ be the clauses of a 3-CNF.

6.1 Proof overview

Fig. 7 shows the structural idea for representing the variables, which we call the *base bundle*. It consists of $(2h-1)n+1$ points arranged as shown in the figure, where h is an appropriate value. The distances between the points are chosen such that the graph $\mathcal{G}$, which connects two points if they are within distance 1, consists of the edges in the figure plus all edges between points in the same level. Thus (except for the intra-level edges, which we can ignore) $\mathcal{G}$ consists of n pairs of paths, one path pair for each variable x_i. The i-th pair of paths represents the variable x_i, and we call it the *x_i-string*. By setting the target size, K, of the problem appropriately, we can ensure the following for each x_i: any feasible solution must use either the top path of the x_i-string or the bottom path, but it cannot use points from both paths. Thus we can use the top path of the x_i-path to represent a TRUE setting of the variable x_i, and the bottom path to represent a FALSE setting. A group of consecutive strings is called a *bundle*. We denote the bundle containing all x_t-strings with $t = i, i+1, \ldots, j$ by *bundle*(i, j).

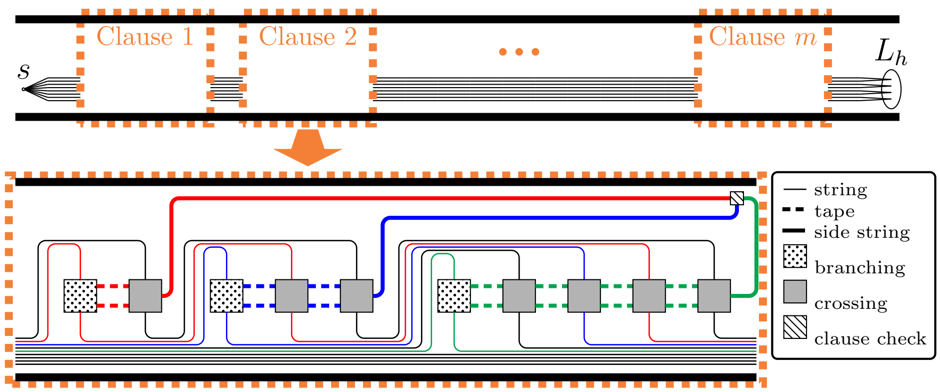

Fig. 15. The overall construction, and the way a single clause is checked. Note that in this figure each string (which actually consists of two paths) is shown as a single curve.

The clause gadgets all start and end in the base bundle, as shown in Fig. 15. The gadget to check a clause involving variables x_i, x_j, x_k, with $i < j < k$, roughly works as follows; see also the lower part of Fig. 15, where the strings for x_i, x_j, and x_k are drawn in red, blue, and green respectively.

First we split off *bundle*$(1, i-1)$ from the base bundle, by letting the top $i-1$ strings of the base bundle turn left. (In Fig. 15 this bundle consists of two strings.) We then separate the x_i-string from the base bundle, and route the x_i-string into a *branching gadget*. The branching gadget creates a branch consisting of two *tapes*—this branch will eventually be routed to the *clause-checking gadget*—and a branch that returns to the base bundle. Before the tapes can be routed to the clause-checking gadget, they have to cross each of the strings in *bundle*$(1, i-1)$. For each string that must be crossed we introduce a *crossing gadget*. A crossing gadget lets the tapes continue to the right, while the string being crossed can return to the base bundle. The final crossing gadget turns the tapes into a *side string* that can now be routed to the clause-checking gadget. The construction guarantees that the side string for x_i still carries the truth value that was selected for the x_i-string in the base bundle. Moreover, if the TRUE path (resp. FALSE path) of the x_i-string was selected to be part of the broadcast set initially, then the TRUE path (resp. FALSE path) of the rest of the

x_i-string that return to the base bundle must be in the minimum broadcast set as well.

After we have created a side string for x_i, we create side strings for x_j and x_k in a similar way. The three side strings are then fed into the clause-checking gadget. The clause-checking gadget is a simple construction of four points. Intuitively, if at least one side string carries the correct truth value—TRUE if the clause contains the positive variable, FALSE if it contains the negated variable—, then we activate a single disk in the clause check gadget that corresponds to a true literal. Otherwise we need to change truth value in at least one of the side strings, which requires an extra disk.

The final construction contains $\Theta(n^4 m)$ points that all fit into a strip of width 40.

In order to simplify our discussion and figures, we scale the input such that a can broadcast to b if their unit disks intersect (or equivalently, if their distance is at most 2).

6.2 Handling strings and bundles

We start the initial bundle directly from the source, and end each string with a disk that intersects the last true and false disk of the given variable, as already seen in Fig. 7. (A TRUE disk is a disk on a TRUE path, a FALSE disk is a disk on a FALSE path.) A minimum-size solution of this bundle for $h = 7$ contains the source disk and true or false disks for each of the 3 strings. In the final construction, once all the clause checks are done and the strings have returned to the bottom bundle, we are going to add some extra levels so that the h-hop restriction does not interfere with the last side strings. (This can be done by for example doubling the maximum distance from s.) The disks of a given level in a bundle lie on the same vertical line, at distance $\frac{1}{2n}$ from each other, so for a bundle containing all the variables, the disk centers on a given level fit on a vertical segment of length 1, and the whole bundle fits in width 3.

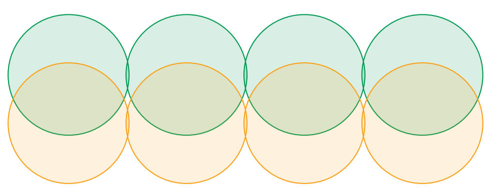

Fig. 16. Disk pairs of a string.

Bundled strings are in lockstep, i.e., a pair of intersecting disks in the bundle that are not in the same string and truth value are on the same level. We call this the *lockstep condition*.

Next, we describe some important aspects of handling strings, bundles and side strings. First, we show that we can do turns with strings in constant horizontal space, and do turns in bundles in polynomial horizontal space. An example of a string turn can be seen in Fig. 17.

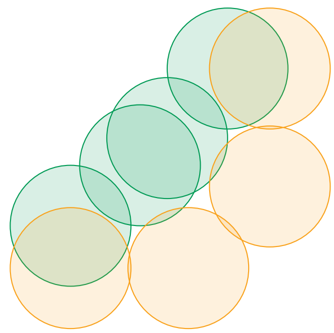

Fig. 17. A turn of 90° in a side string or string outside a bundle using constant horizontal space.

This turning operation can be used on the top string of a bundle to "peel" off strings one by one and unify them later in a new bundle, see Fig. 18. This is how we can split and turn a bundle: we peel and turn the strings one by one. Notice that the lockstep condition is upheld both in the bottom and top bundle. It requires $O(n)$ extra horizontal space and $O(n^2)$ disks to split a bundle with this method.

If we were to return the strings one by one to the bottom bundle without correction as depicted in Fig. 15, the returning strings would be in a level disadvantage compared to the bottom bundle, so the new bundle would violate the lockstep condition. To avoid this issue, we use a correction mechanism. We have some room to squeeze bundle levels horizontally. The largest horizontal distance between neighboring levels is 2; for the smallest distance, we need to make sure that a disk does not intersect other disks from neighboring levels other than the disks in the same string with the same truth value. So the horizontal distance has to be at least $2\sqrt{1-\left(\frac{1}{4n}\right)^2} < 2 - \frac{1}{15n^2}$. Thus, if we have $15n^2$ compressed levels in a bundle, then they take up the same horizontal space as $15n^2 - 1$ maximum distance levels.

A detour of a string (peeling off, going through a gadget, returning to the bottom bundle) requires a constant number of extra levels to achieve, we can compensate for this with the addition of a polynomial number of extra disks. Before a string peels off from the top bundle downward to rejoin the bottom bundle, we add $15n^2k$ compressed levels to the top bundle and $(15n^2-1)k$ maximum distance levels to the bottom bundle, if the total number of extra levels added by turning up, going through the gadget and turning down is k.

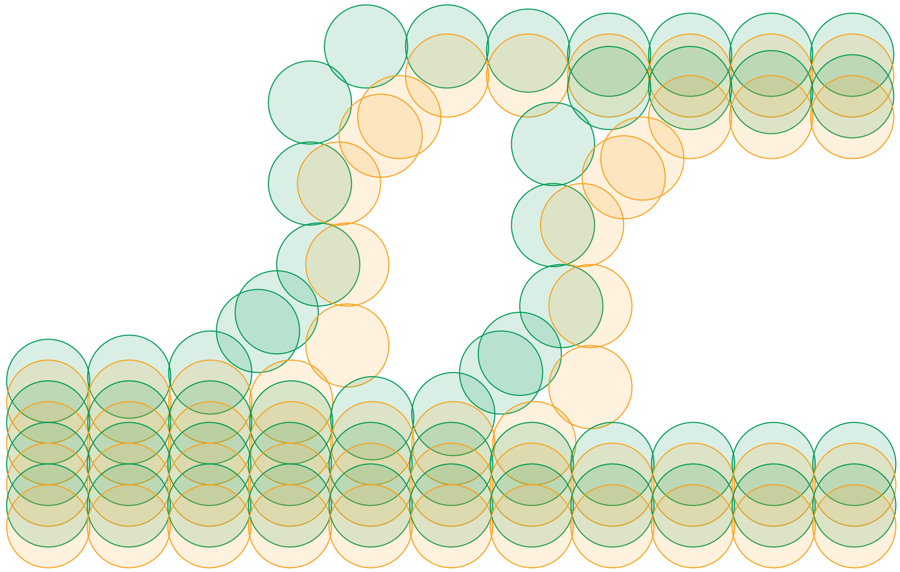

Fig. 18. Splitting the top 2 strings off a bundle of 4 strings: we peel the top layers one by one.

This ensures that the lockstep condition is upheld in the bottom bundle after the return of this string. For each string that leaves the bottom bundle and later returns, we use this correction mechanism. Overall, this correction mechanism is invoked a polynomial number of times, so requires a polynomial number of disks.

6.3 Tapes

Our tapes consist of tape blocks: a tape block is a collection of three disks, the centers of which lie on a line at distance ϵ apart – so it is isometric to the old connector blocks A, B, C and D for the case "n"$= 2$ (see Section 4). Denote the three disks inside a tape block T^k by δ_1^k, δ_2^k and δ_3^k. We can place multiple such blocks next to each other to form a tape. An example is depicted in Fig. 19.

The tapes always connect blocks in which disks have truth values assigned, e.g., the end of a string or disks of a gadget block. Denote the starting true and false disks by F and the ending true and false disks by G. We say that a set of tape blocks $T^1, T^2, \ldots, T^p$ forms a tape from F to G if it satisfies the following conditions.

- In the first block, δ_1^1 intersects both the true and false disk(s) of F, δ_2^1 intersects the true disk(s) of F, and δ_3^1 is disjoint from both the true and false disk(s).
- δ_i^k intersects the disk δ_j^{k+1} if and only if $j \leqslant i$ ($k = 1, \ldots, p-1$).

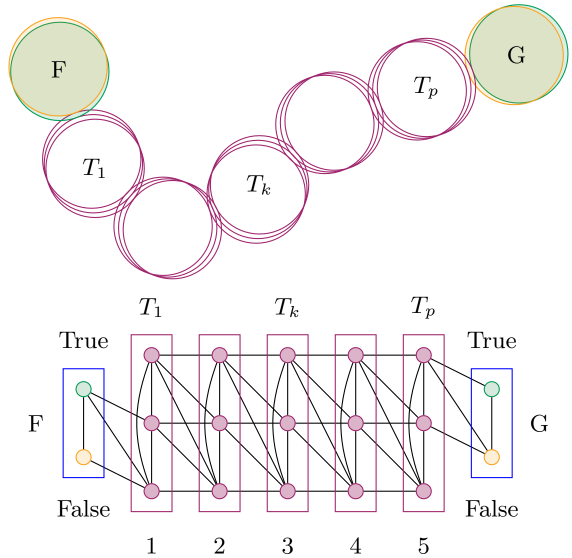

Fig. 19. Tape blocks connecting two true-false disk pairs and the corresponding subgraph.

- In the last block, δ_1^p is disjoint from G, δ_2^p intersects the false disk(s) of G, and δ_3^p intersects both the true and false disk(s).
- Non-neighboring tape blocks are disjoint, F is disjoint from all blocks except the first, and G is disjoint from all blocks except the last.

We would like to examine the set of disks that are used in a minimum broadcast set from a tape.

Lemma 12. *Let T be a tape from F to G that has p tape blocks. Every h-hop broadcast set contains at least $p-1$ disks from the tape. If a broadcast set contains exactly $p-1$ disks, then the truth value of F is is at least the truth value of G, i.e., it cannot happen that the active disks in F are all false disks and the active disks in G are all true disks.*

Proof. Let the tape blocks be $T^1, T^2, \ldots, T^p$. If there are at most $p-2$ active disks, then there are at least two empty blocks. These blocks have to be neighboring, otherwise a point in between the two blocks is impossible to reach from the source. Let these blocks be T^k and T^{k+1}. All disks in T_k must be reached through the blocks $F, T^1, \ldots, T^{k-1}$. Specifically, δ_3^k has to be reached. The shortest path to this point from any F-disk requires at least k tape disks. Similarly, the shortest path from any G-disk to δ_1^{k+1} requires at least $t-k$ disks. Overall, at least t active disks of the tape are required to reach these disks – this is a contradiction.

If the tape contains $t-1$ active disks, then both F and G must contain an active disk, otherwise there would be a component inside the tape that is not connected to the source. Suppose for contradiction that the active disks of F are false and the active disks of G are true. There is at least one tape block that has no active disk; let T^k be such a block, where k is as small as possible. Since δ_2^k has to be covered, it has to be reached either from F or G.

Suppose that δ_2^k is reached through F; this requires k active disks from the tape blocks. We have only $p-1-k$ active disks for the rest of the $p-k$ blocks $T^{k+1}, \dots, T^p$, so there has to be another empty tape block, T^ℓ ($\ell > k$). As previously demonstrated, we cannot have non-neighboring empty blocks, so the other empty block is T^{k+1}. This means that δ_3^k also has to be dominated from the left side, the shortest path to which requires $k+1$ active tape disks from any false disk of F. This leads to an additional empty block among $T^{k+1}, \dots, T^p$. But as shown above, there can be at most one such block (T^{k+1}) – we arrived at a contradiction. The same argument works for the case when δ_2^k is reached from G. ☐

6.4 Gadgets and their connection to tapes and strings

Crossing and branching gadgets. Our crossing gadget and our branching gadget are almost identical to the one used in the W[1]-hardness proof of CDS-UDG. This gadget can be used to transmit information both horizontally and vertically – this is exactly what we need. Since we only need to transmit truth values, we take the gadget for "n"$=2$, resulting in X blocks with $2 \cdot 2$ and Y -blocks with $2 \cdot 2 + 1$ disks. The only change we make in the crossing gadget is that we swap the X_1 and X_2 blocks.

For the branching gadget, we modify some offsets so that we can transmit the vertical truth value on the right side of our gadget. For this purpose, we redefine the offsets in the following right side X-blocks.

$$\text{offset}(X_3(j)) = (-\iota_2(j), -j) \qquad \text{offset}(X_4(j)) = (\iota_2(j), -j)$$

In case of these horizontal connections, we say that a disk $X_k(j)$ from the block X_k is a true disk if $\iota_1(j) = 2$ and a false disk if $\iota_1(j) = 1$. Similarly, for vertical connections, a disk $X_\ell(j)$ is a true disk if $\iota_2(j) = 2$ and a false disk if $\iota_2(j) = 1$.

Connecting gadgets with tapes and strings. When connecting branching and crossing gadgets or two crossing gadgets with tapes horizontally, we are going to add a tape that goes from the X_4 block of the left gadget to the X_7 block of the right gadget, and a tape that goes from the X_8 block of the right gadget to the X_3 block of the left gadget. Note that in the W[1]-hardness proofs, we used the same strategy with tapes consisting of only one block. In this case, we place the first and last block of each tape at the same location as the connector block in the proof of Theorem 6, and use some tape blocks in between these, the number of which will be specified later. Note that this placement gives us a tape that is consistent with the definition of true and false disks in the X-blocks.

In order to connect strings and side strings to the gadgets, we use both tapes and parent-leaf pairs. Fig. 20 depicts a connection to a crossing gadget from the top and bottom.

We need to connect both "sides" of the string: on the top, we use a tape from X_2 to the last string block, and a tape from the last string block to X_1. Moreover, in order to make sure that all the disk pairs of the strings are in use, and the connection is not maintained through the tapes, we create a short path to the gadget with some disks that are guaranteed to be in the solution. This path consists of parent and leaf disk pairs, where all the parents will be inside a canonical solution – we used this technique before inside the gadgets to ensure gadget connectivity. The string exits the gadget similarly. Note that the shortest path through the gadget from the string end on the top to the string end on the bottom has length 18, and its internal vertices are all parent disks, a disk from X_1' and a disk from X_5'; the paths using any of these tapes are longer.

We use the same type of connection to connect side strings to the right side of the last crossing gadget (or to the branching gadget, if the current clause contains the first variable). The complete gadget together with the connections and string turns fits in 50 units of vertical space. (Recall that all distances have been scaled by a factor of two, so that we have unit radius disks.)

We briefly return to the tape pairs that connect neighboring blocks. We need to make sure that the tapes do not provide a shortcut – we want the shortest path from source to the last level h to be through string blocks, and to go through gadgets as discussed above. When choosing a tape length, we also need to bridge the distance between neighboring gadgets. Note that this amount can be polynomial in n because of the correction mechanism for strings. We add a small detour to make sure that the shortest path to a gadget that uses a tape is longer than the shortest path that uses only the string that enters the gadget. It is easy to see that there is enough place for such a detour: taking twice the amount of blocks that would be necessary to cover the distance is enough. A tape connection between neighboring blocks is depicted in Fig. 21. (Note that these tapes need no additional vertical space: they fit easily in the 18 units of vertical space between the gadgets.)

The clause check gadget. The clause check gadget is very simple, it contains four well-placed disks: one at the end of each of the three side strings, and one disk that only intersects the three other clause check disks. We turn the three side strings towards their corresponding disks so that the side strings do not interfere with each other. Among the six last disks at the end of the three side strings only the ones corresponding to the literals of this clause intersect the gadget. The rest of the side string disks are disjoint from the gadget. See Fig. 22 for an example of checking $(x_2 \vee x_3 \vee \bar{x}_5)$. The vertical space required is less than 20 units.

Our complete construction can fit in 80 units of vertical space. Ten units can accommodate the lower bundle and turning strings up and down from it;

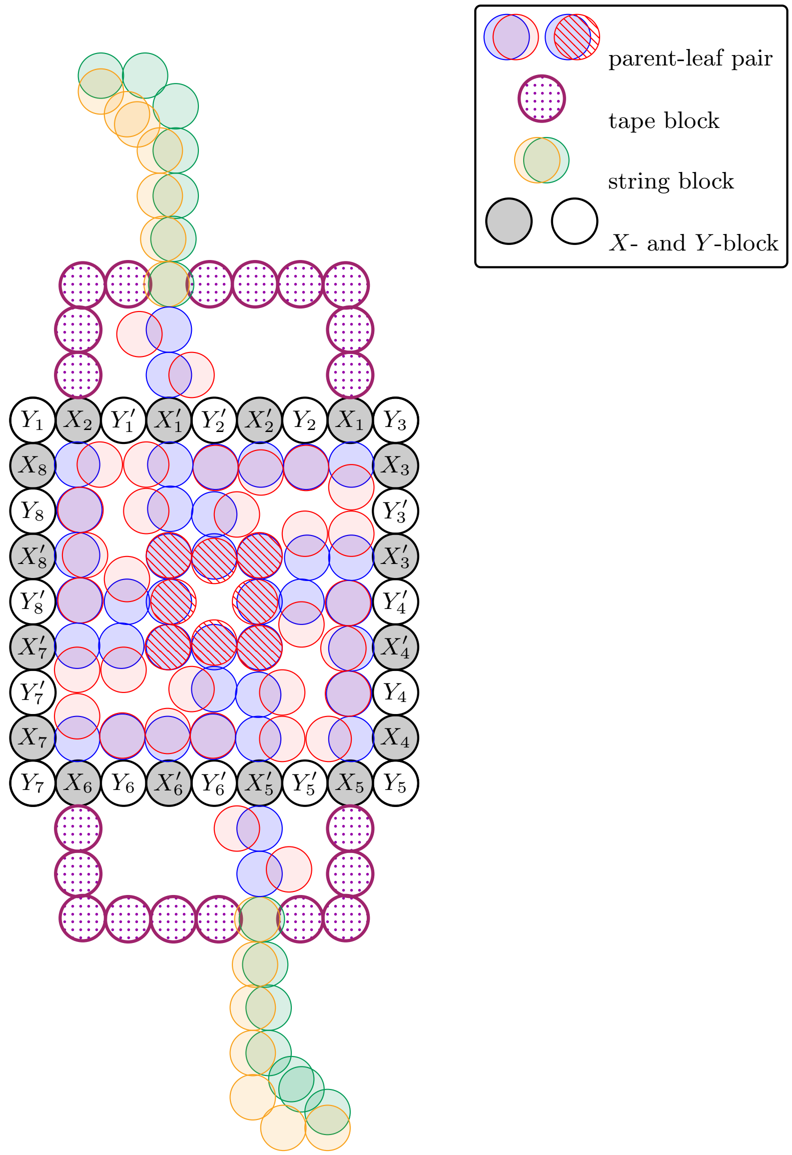

Fig. 20. Connecting to a crossing or branching gadget from the top and bottom.

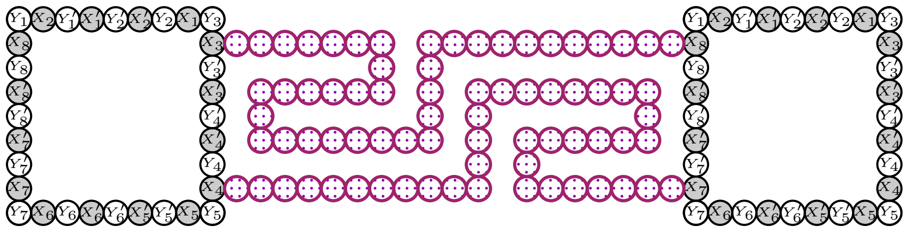

Fig. 21. Connecting neighboring gadgets with tapes.

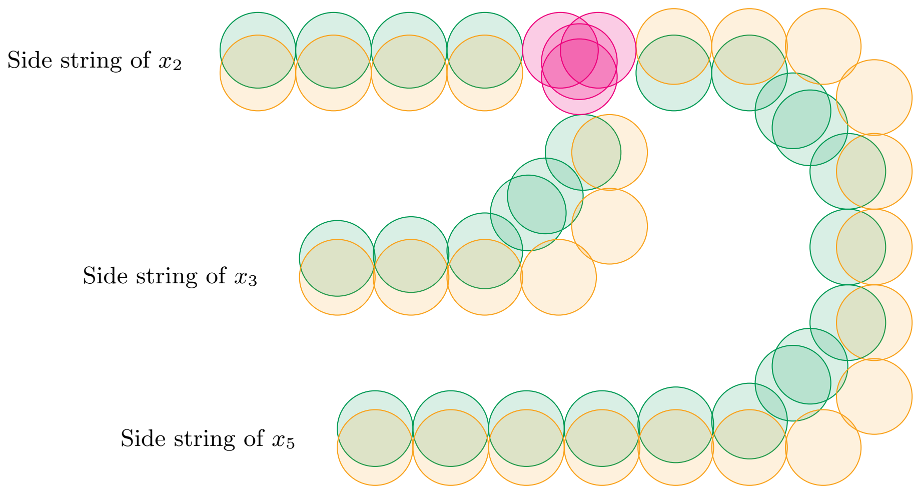

Fig. 22. Clause check gadget for the clause $(x_2 \vee x_3 \vee \bar{x}_5)$.

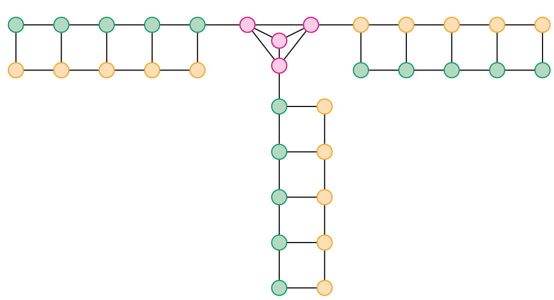

Fig. 23. The subgraph spanned by the disks of a clause check gadget and its surroundings.

50 units of vertical space can accommodate the branching and crossing gadgets, along with their connections and tapes. We need ten units for the bundle that goes above the gadgets (along with the string turns), and finally 20 more for the side strings and the clause check gadget. Recall that we did a scaling by two to switch to the intersection model of broadcasting. In the original model of broadcasting, the construction occupies 40 units of vertical space.

In case of a satisfiable formula, we can choose the disks in each side string that correspond to the value of the variable, and choose a disk from the clause check gadget that intersects a true literal (at least one of the literals is true in the clause).

This lemma describes the usage of the clause check gadgets and the side strings.

Lemma 13. *Let ϱ be the number of true-false disk pairs (blocks) in the three side strings that correspond to a particular clause checking gadget. An h-hop broadcast set contains at least $\varrho + 1$ disks from the three side strings and the clause check gadget. Moreover, if an h-hop broadcast set has exactly $\varrho + 1$ actives among these disks, then the truth values chosen at the beginning of the side strings satisfy the clause.*

Proof. We prove the following claim first.

> *Claim.* A side string cannot contain two empty blocks.
>
> *Proof of claim.* Suppose that U_k and U_ℓ are two empty side string blocks. If they are not neighboring, then a disk between them is unreachable from the source. So $\ell = k + 1$. Consequently, both disks of U_k are dominated from the start of the side string, and both disks of U_{k+1} are dominated from the end. Since the side string has length at more than four, either $k > 2$ or $k < p - 1$. Suppose $k > 2$, the other case is similar. The only way to reach both disks in U_k is to have both of the disks in U_{k-1} active. Since U_{k-1} is also reached from the left, there is an active disk in U_{k-2}; let its truth value be v. So we can deactivate the disk in U_{k-1} of value $\neg v$ and activate the disk in U_k of value v. This way every disk that has been dominated remains dominated, and the number of active disks does not increase. (Note that we do not need to worry about exceeding h hops since h will be chosen large enough to not interfere with side strings.) □

Now we show that every h-hop broadcast set includes at least $\varrho + 1$ disks from these side strings and the clause check gadget. Suppose there is a side string of p blocks that contains an empty block U_k, and let v be the truth value of the disk in the last block (the one intersecting the clause check gadget). Suppose $2 \leqslant k \leqslant p - 1$; a small variation of the argument applies to the cases $k = 1$ and $k = p$. In U_{k-1}, the disk of value $\neg v$ has to be reached from the beginning of the string — the shortest path requires at least $k - 2$ active disks in $U_1, \ldots, U_{k-2}$. In U_{k+1}, the disk of value $\neg v$ has to be reached through the clause check gadget; this requires that the clause check disk corresponding to this side string is active, and there are at least $p-k$ side string actives from $U_{k+1}, \ldots, U_p$, since we also need to change truth value along the way. Additionally, the disk of value $\neg v$ in U_k has to be reached from one of the neighboring blocks, requiring $U_{k-1}(\neg v)$ or $U_{k+1}(\neg v)$ to be active. Overall, either a side string does not contain an empty block (so it has at least p disks), or we needed $k-2+(p-k)+1+1=p$ active disks from the side string and the corresponding clause check disk. Moreover, at least one of the three side strings needs to connect the middle point of the clause check gadget to the source: the shortest path through a side string of p blocks has $p+1$ inner vertices, since it has to include one disk from each block of the side string and the clause check disk corresponding to this side string. Consequently, we need at least $\varrho+1$ active disks.

Finally, we need to show that if the assignment at the beginning of the side strings does not satisfy the clause (all literals are false), then we need at least $\varrho+2$ active disks. A similar argument shows that the string that reaches the clause check gadget must have one extra active disk. □

6.5 Reduction from 3-SAT

Let β be the number of branchings, let γ be the number of crossings, let ξ be the number of disk pairs inside strings and side strings, and let τ be the number of tape blocks in our construction. We examine the disks that are necessarily part of a minimum broadcast set if the formula is satisfiable. It will be apparent that a solution of the same size cannot exist if the formula is not satisfiable.

We include all the disks from the strings and side strings that correspond to the value given to the variable, altogether ξ disks. Add the disks from the gadgets: the blue parent disks inside and one disk from each X-block, altogether $52(\beta+\gamma)$ disks. The branching and crossing gadget connections require four blue parent disks outside the gadget at the top and bottom connection, and two more disks on the last gadget (one per branching), so we require $4(\beta+\gamma)+2\beta$ for connections. In each tape we include one disk from all of its blocks except one. The number of tapes that connect neighboring gadgets is 2γ, and we also use 2 tapes per string- gadget connection, so we have $4(\beta+\gamma)+2\beta$ such tapes. Thus, the number of tape disks in a solution is $\tau-(2\gamma+4(\beta+\gamma)+2\beta)$. Finally, we use one disk to cover each clause check gadget, overall m disks. (Recall that m is the number of clauses.)

In case of a satisfiable formula, the total number of disks required for a canonical broadcast set is

$$\begin{aligned} &\xi+52(\beta+\gamma)+4(\beta+\gamma)+2\beta+\tau-\big(2\gamma+4(\beta+\gamma)+2\beta\big)+m \\ =\,&\xi+52\beta+50\gamma+\tau+m. \end{aligned}$$

Theorem 10. *There is a minimum h-hop broadcast set of cost $C=\xi+52\beta+50\gamma+\tau+m$ if and only if the original 3-CNF formula is satisfiable.*

Proof. As we demonstrated previously, if the formula is satisfiable, then there is an h-hop broadcast set of the given size. We need to show that if there is an h-hop broadcast set of this size, then the formula is satisfiable. Take a minimum h-hop broadcast set. First, we know that the shortest path to the string ending disks requires exactly h hops, and the only path of this length includes all blocks of the string in question, plus the shortest way through the gadgets in which this string is involved. It is easy to check that the shortest way through a gadget from the string end on the top to the string end on the bottom uses only blue disks, and one disk from X_1' and X_5' each. Without loss of generality we can suppose that the h-hop broadcast set restricted to each gadget is canonical by the analogue of Lemma 9. Let $t = 2\gamma + 4(\beta+\gamma) + 2\beta$ be the number of tapes in the construction. A minimum h-hop broadcast set must include at least $\tau - t$ tape disks, and for each clause i, at least $\varrho_i + 1$ active disks as shown by Lemmas 12 and 13, where ϱ_i is the number of blocks in the three side strings that correspond to a clause. So a minimum h-hop broadcast set does indeed require at least C disks. An h-hop broadcast set of this size that is canonical when restricted to each gadget also means that the truth value carried by a string before entering a gadget is the same as the truth value carried after exiting the gadget. Similarly, the truth values are transferred between neighboring gadgets connected by a tape pair: this can be seen by applying Lemma 12 for both tapes. And finally, all clauses must have a true literal at the beginning of at least one of the corresponding side strings by Lemma 13. Since the disk choice at the beginning of a side string is forced to comply with the corresponding string, it follows that the truth values defined by the strings satisfy the formula. □

Our construction can be built in polynomial time – note that the coordinates of each point can be represented with $O(\log n)$ bits, since a precision of c/n^4 is sufficient. We have successfully reduced 3-SAT to the h-hop broadcast problem in a strip of width 40. Since the problem is trivially in NP, this concludes the proof of Theorem 9.

7 Conclusion

We studied the complexity of the broadcast problem in narrow and wider strips. For narrow strips we obtained efficient polynomial algorithms, both for the non-hop-bounded and for the h-hop version, thanks to the special structure of the problem inside such strips. On wider strips, the broadcast problem has an $n^{O(w)}$ algorithm, while the h-hop broadcast becomes NP-complete on strips of width 40. With the exception of a constant width range (between $\sqrt{3}/2$ and 40) we characterized the complexity when parameterized by strip width. We have also proved that the planar problem (and, similarly, CDS-UDG) is W[1]-hard when parameterized by the solution size. The problem of finding a planar h-hop broadcast set seems even harder: we can solve it in polynomial time for $h = 2$ (see Appendix A) but already for $h = 3$ we know no better algorithm than brute force. Interesting open problems include:

- What is the complexity of planar 3-hop broadcast? In particular, is there a constant value t such that t-hop broadcast is NP-complete?

- What is the complexity of h-hop broadcast in planar graphs?

References

1. 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)
2. 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)
3. Berg, M.d., Cheong, O., Kreveld, M.v., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer-Verlag, 3rd edn. (2008)
4. Breu, H.: Algorithmic aspects of constrained unit disk graphs. Ph.D. thesis, University of British Columbia (1996)
5. Cabello, S., Jejčič, M.: Shortest paths in intersection graphs of unit disks. Comput. Geom. 48(4), 360–367 (2015)
6. Chazelle, B., Guibas, L.J.: Fractional cascading: I. A data structuring technique. Algorithmica 1(1), 133–162 (1986)
7. 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)
8. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer (2015)
9. 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)
10. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks 1(3), 195–207 (1971)
11. Edelsbrunner, H., Guibas, L.J., Stolfi, J.: Optimal point location in a monotone subdivision. SIAM J. Comput. 15(2), 317–340 (1986)
12. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series, Springer (2006)
13. Fuchs, B.: On the hardness of range assignment problems. Networks 52(4), 183–195 (2008)
14. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
15. Kratsch, D., Stewart, L.: Domination on cocomparability graphs. SIAM J. Discrete Math. 6(3), 400–417 (1993)
16. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. 11(2), 329–343 (1982)
17. 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)
18. Marx, D., Pilipczuk, M.: Optimal parameterized algorithms for planar facility location problems using Voronoi diagrams. In: ESA, Proceedings. pp. 865–877 (2015)
19. Masuyama, S., Ibaraki, T., Hasegawa, T.: The computational complexity of the m-center problems on the plane. IEICE Transactions 64(2), 57–64 (1981)
20. 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)

A Planar 2-hop broadcast

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

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 $Q := \{q_1, \dots, q_m\}$ 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 $\{i, \dots, j\}$ if $i \leqslant j$, and we define $[i, j]$ to be the set of indices $\{i, \dots, m, 1, \dots, j\}$ 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

$$\text{OPT} = \min_{i,j} \big(A(i, j) + A(j + 1, i - 1)\big). \tag{4}$$

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

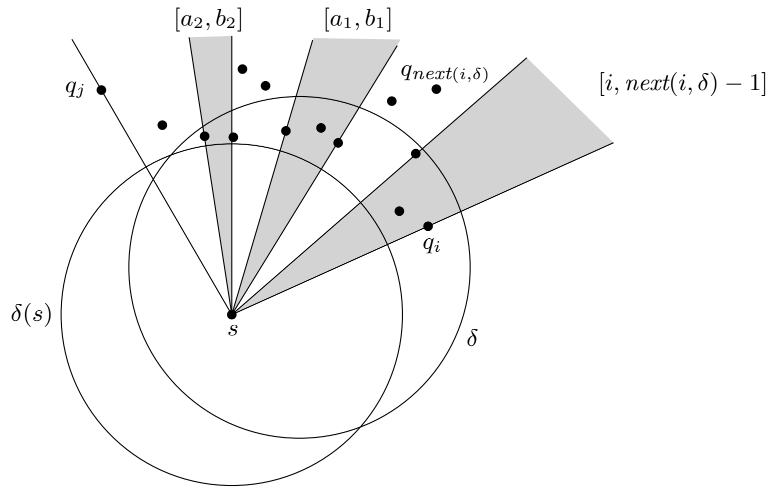

Fig. 24. Definition of the intervals $[a_i, b_i]$

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. 24 where the angular ranges containing covered points are indicated in gray. We can thus define a set of maximal intervals that together form $\delta \cap Q[i, j]$:

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

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:

$$A(i, j) = \begin{cases} 1 & \text{if } i = j \\ 1 + \min \left\{ A(next(i), j), \min\limits_{\substack{\delta \in \Delta_i \\ (a,b) \in \mathfrak{I}(i,j,\delta)}} (A(next(i, \delta), a) + A(b, j)) \right\} & \text{otherwise} \end{cases} \tag{5}$$

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 12. *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 $q \in Q$, 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 $ray(q)$ exists $\mathcal{U}(D)$. 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 $\sigma(D) := \langle pred(q_1), \dots, pred(q_m)\rangle$. 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 13. *In a 2-hop broadcast set D, the boundary sequence $\sigma(D)$ has no cyclic subsequence $\cdots p \cdots p' \cdots p \cdots p'$ 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 $\cdots p \cdots p' \cdots p \cdots p'$ cannot occur in σ. □

Lemma 14. *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. 25. It is easy to see that $\alpha(\gamma) \leqslant 240°$. 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 $\delta(s)$,

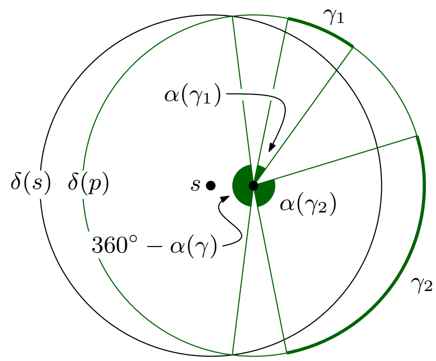

Fig. 25. Illustration for the proof of Lemma 14.

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 $\delta(s) \setminus \delta(p)$. □

We are now ready prove the correctness of our algorithm.

First consider Equation (4). It is clear that

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

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 $\{q \in Q : \mathit{pred}(q) = p\} = Q[i,j]$. Then $A(i,j) = 1$ and there is a covering of $Q[j+1, i-1]$ with $|D| - 1$ disks. Hence, $\min_{i,j} \left(A(i,j) + A(j+1, i-1)\right) \leqslant \text{OPT}$ 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 $\{q \in Q : \mathit{pred}(q) = p\} = Q[i_1, j_1] \cup Q[i_2, j_2]$. 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 13. Hence, $(A(i_1, j_2) + A(j_2+1, i_1-1)) \leqslant \text{OPT}$.

Next, we prove that the recursive formula (5) 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 $\delta \in \varDelta$ that can cover q_i. Now consider the case $i \neq j$. First we note that

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

Indeed, there is a disk covering $Q[i, \mathit{next}(i)-1]$ by definition of $\mathit{next}(i)$ and we can cover $Q[\mathit{next}(i), j]$ by $A(\mathit{next}(i), j)$ disks by induction. Similarly, the definition of

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

To prove the reverse, let D be a minimum-size 2-hop broadcast for $Q[i,j]$ and let $p := \mathit{pred}(q_i)$. If p appears in the covering of $Q[i,j]$ only once, then $A(i,j) = 1 + A(\mathit{next}(i), j)$. Otherwise p appears twice by Lemma 14. 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 13, the coverings of $Q[\mathit{next}(i,\delta), a]$ and $Q[b,j]$ are disjoint in D. Hence, $1 + (A(\mathit{next}(i,\delta), a) + A(b,j)) \leqslant |D|$. We conclude that

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

It remains to analyze the running time. The algorithm works by first computing $\mathit{next}(i)$, $\mathit{next}(i,\delta)$ 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 (5) 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 Equation (4) takes $O(n^2)$m 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. □