A $(4 + \epsilon)$-Approximation for the Minimum-Weight Dominating Set Problem in Unit Disk Graphs

Thomas Erlebach[1] and Matúš Mihalák[2]

[1] Department of Computer Science, University of Leicester, England
[2] Institute of Theoretical Computer Science, ETH Zurich, Switzerland

Abstract. We present a $(4 + \epsilon)$-approximation algorithm for the problem of computing a minimum-weight dominating set in unit disk graphs, where ϵ is an arbitrarily small constant. The previous best known approximation ratio was $5+\epsilon$. The main result of this paper is a 4-approximation algorithm for the problem restricted to constant-size areas. To obtain the $(4 + \epsilon)$-approximation algorithm for the unrestricted problem, we then follow the general framework from previous constant-factor approximations for the problem: We consider the problem in constant-size areas, and combine the solutions obtained by our 4-approximation algorithm for the restricted case to get a feasible solution for the whole problem. Using the shifting technique (selecting a best solution from several considered partitionings of the problem into constant-size areas) we obtain the claimed $(4 + \epsilon)$-approximation algorithm. By combining our algorithm with a known algorithm for node-weighted Steiner trees, we obtain a 7.875-approximation for the minimum-weight connected dominating set problem in unit disk graphs.

1 Introduction

A subset $D \subseteq V$ of the vertices of an undirected graph $G = (V, E)$ is called a *dominating set* if every vertex in V is contained in D or has a neighbor in D. A vertex in D is called a *dominator*, and we say that a dominator *dominates* itself and all its neighbors. The *minimum dominating set problem* (MDS) is to compute a dominating set of smallest size. MDS belongs to the classical $\mathcal{NP}$-hard optimization problems listed in the book of Garey and Johnson [7]. MDS for general graphs is equivalent to the *set cover* problem, and can thus be approximated within a factor of $O(\log n)$ for graphs with n vertices using a greedy algorithm (see, e.g., [16]), but no better unless all problems in $\mathcal{NP}$ can be solved in $n^{O(\log \log n)}$ time [6]. If every vertex of the input graph is associated with a weight, the *minimum-weight dominating set problem* (MWDS) is to compute a dominating set of minimum weight. Approximation ratio $O(\log n)$ can also be achieved for the weighted set cover problem and thus for MWDS [7]. The variants of the problems where the dominating set is asked to be connected in the input graph are called, in an obvious way, the *minimum connected dominating set* *problem* (MCDS) and the *minimum-weight connected dominating set problem* (MWCDS), respectively. The best known approximation ratio for MWCDS in general graphs is $O(\log n)$ as well [8].

We consider the problem of computing a minimum-weight (connected) dominating set in *unit disk graphs*. A unit disk graph is a graph where every vertex is associated with a disk of unit radius in the plane and there is an edge between two vertices of the graph if the two corresponding disks intersect. These problems are $\mathcal{NP}$-hard already for the unweighted case [4, 12]. We are thus interested in approximation algorithms. An algorithm for MDS (or MWDS) is called a *ρ-approximation algorithm*, and has *approximation ratio ρ*, if it runs in polynomial time and always outputs a dominating set whose size (or total weight) is at most a factor of ρ larger than the size (or total weight) of the optimal solution. The definitions for MCDS and MWCDS are analogous. A *polynomial-time approximation scheme* (PTAS) is a family of approximation algorithms with ratio $1+\varepsilon$ for every constant $\varepsilon > 0$.

Constant-factor approximation algorithms for MDS and MCDS in unit disk graphs were given by Marathe et al. [13]. For MDS in unit disk graphs, a PTAS was presented by Hunt et al. [11], based on the shifting strategy [2, 9]. These algorithms, however, do not extend to the weighted version. In particular, the PTAS is based on the fact that the optimal dominating set for unit disks in a $k \times k$ square has size at most $O(k^2)$ and can thus be found in polynomial time using complete enumeration if k is a constant. In the weighted case, there is no such bound on the size of an optimal (or near-optimal) solution, as an optimal solution may consist of a large number of disks with tiny weight. For MCDS in unit disk graphs, a PTAS was presented in [3]. For unit disk graphs with bounded density, asymptotic fully polynomial-time approximation schemes (with running time polynomial in $\frac{1}{\varepsilon}$ and in the size of the input, but achieving ratio $1+\varepsilon$ only for large enough inputs) were presented for MDS and MCDS in [15].

The first constant-factor approximation algorithms for MWDS and MWCDS in unit disk graphs were given by Ambühl et al. [1], with approximation ratios 72 and 89, respectively. Huang et al. [10] presented approximation algorithms with approximation ratio $6+\epsilon$ and $10+\epsilon$, respectively. Currently the best approximation algorithms for MWDS is due to Dai and Yu [5], with approximation ratio $5+\epsilon$. Zou et al. [17] present an approximation algorithm with ratio $2.5\rho < 3.875$ for the node-weighted Steiner tree problem in unit disk graphs, where $\rho = 1+\frac{\ln 3}{2}$ is the best known approximation ratio for the classical Steiner tree problem [14]. This result can be used to connect a dominating set by adding nodes of weight at most 2.5ρ times the weight of an optimal connected dominating set, yielding the currently best approximation ratio of 8.875 for MWCDS.

Our Results. We present a $(4+\epsilon)$-approximation algorithm for MWDS in unit disk graphs. Our algorithm is based on several ideas of previous constant-factor approximation algorithms for the problem [1, 10]. We partition the plane into areas of size $K \times K$, where K is an arbitrary constant. For each of these areas we consider the following subproblem: find a minimum-weight set of disks that dominate all disks that have a center in the area. The union of feasible solutions for each subproblem yields a dominating set for the original problem. Using the shifting technique as presented in [10], the loss in the approximation factor is only $(1 + O(1)/K)$, i.e., if the solution for every subproblem is a ρ-approximation, then, using the shifting technique, the (best) combination of the solutions is a $(\rho + O(1)/K)$-approximation for the original problem. Thus, for any constant ϵ, one can set K such that the obtained solution is a $(\rho + \epsilon)$-approximation. We present a 4-approximation algorithm for the subproblem, which thus leads, using the shifting technique and setting K appropriately, to a $(4 + \epsilon)$-approximation algorithm. We note that Huang et al. [10] presented a 6-approximation for the subproblem, and Dai and Yu [5] a 5-approximation for the subproblem. Connecting the dominating set computed by our algorithm using the node-weighted Steiner tree algorithm of Zou et al. [17], we obtain a 7.875-approximation for MWCDS, which improves the previously best approximation ratio of 8.875.

We note that independently from our work, Zou et al. [18] have also obtained a $(4 + \epsilon)$-approximation algorithm for MWDS.

The Problem as a Covering Problem. We assume an instance of the problem is given by a set $\mathcal{D}$ of n weighted unit disks in the plane, where every disk $d \in \mathcal{D}$ has radius 1 and weight w_d. We denote by $\mathcal{C}$ the centers of the disks in $\mathcal{D}$. Also, for a set of disks $X \subseteq \mathcal{D}$, we denote by $w(X)$ the total weight of disks in X, i.e., $w(X) = \sum_{d \in X} w_d$.

In the following we consider the problem as a covering problem – for a set $\mathcal{C}$ of centers of disks $\mathcal{D}$, every disk of the same radius, find a minimum-weight set $\mathcal{D}' \subseteq \mathcal{D}$ of disks the union of which contains all points in $\mathcal{C}$. If a disk d contains point p, we say that the disk d *covers* point p, and that p *is covered by* d. It is not difficult to see that the original problem and this covering problem are in fact equivalent – a dominating set for input $\mathcal{D}$ of unit disks induces a solution for the covering problem given by disks of radius 2 with centers identical to centers $\mathcal{C}$, and a solution to the covering problem induces a dominating set for the original problem. Thus, given an instance of MWDS, we can consider the equivalent covering problem with disks of radius 2. Scaling the setting down by a factor of 2 (i.e., dividing the coordinates of the center of every disk by 2, and considering disks of unit radius) we obtain an instance of the covering problem with unit disks. From now on we assume we have performed such a modification to the setting, and our goal is to find a minimum-weight subset of unit disks $\mathcal{D}$ that cover all points $\mathcal{C}$.

Structure of the paper. We first present the general approach to solving the covering problem by considering covering subproblems induced by constant-size squares in Sect. 2. In Sect. 3 we present our 4-approximation algorithm for the covering subproblem. We conclude the paper in Sect. 4.

2 General Algorithm for the Covering Problem

The general algorithm follows the approach of Huang et al. [10]. First, we partition the plane into squares of size $\mu \times \mu$, where $\mu = \frac{\sqrt{2}}{2}$. The square S_{ij},

$i, j \in \mathbb{Z}$, contains points with coordinates (x, y), where $i \cdot \mu \leq x < (i+1) \cdot \mu$, and $j \cdot \mu \leq y < (j+1) \cdot \mu$. We say that a disk $d \in \mathcal{D}$ *is from* square S_{ij}, if the center of disk d lies in S_{ij}. For a square S_{ij} we denote by $\mathcal{D}_{ij}$ the disks from S_{ij}, and by $\mathcal{C}_{ij}$ the centers of disks in S_{ij}. Notice that the size of squares is chosen such that any disk from square S_{ij} contains the whole square S_{ij}, and thus covers all centers $\mathcal{C}_{ij}$.

Second, we consider the squares S_{ij}, $i, j \in \mathbb{Z}$, in groups, each group consisting of $k \times k$ squares, $k \geq 1$. We call such a group of squares a *block*. Formally a block $B_{a,b}$ consists of squares S_{ij}, $a \cdot k \leq i < (a+1) \cdot k$, and $b \cdot k \leq j < (b+1) \cdot k$. We say that a disk $d \in \mathcal{D}$ *is from* block B, if the center of d lies in B. We denote by $\mathcal{D}_B$ the set of disks from block B, and by $\mathcal{C}_B$ the set of centers from block B.

For each block B, we consider the following *covering subproblem induced by block B*: find a minimum-weight set of disks in $\mathcal{D}$ that covers the centers $\mathcal{C}_B$. Let X_B denote a feasible solution to the covering subproblem for block B. Clearly, the union X of the solutions X_B for every block B is a feasible solution for the whole covering problem. Observe that only disks from B and disks with centers at distance at most one from B need to be considered for X_B. Thus, only disks close to the boundary of every block can be part of solutions to more than one block. Consider now an optimum solution OPT for the covering problem. For an appropriate choice of the origin of the coordinate system (which causes different positioning of the grid formed by the blocks), a substantial part of OPT, in terms of the weight of disks, is formed by disks in the area formed by the "central" parts (i.e., not close to the boundary) of nonempty blocks. Thus, we can use the shifting technique to try out different choices for the origin and construct a good feasible solution X for the covering problem: Consider the $k/4$ partitionings in blocks induced by the origin set to $(p \cdot (4\mu), p \cdot (4\mu))$, $p = 0, 1, \ldots, k/4$ (observe that for every such choice of the origin the partitioning into squares S_{ij}, $i, j \in \mathbb{Z}$, is the same, just every square has now different subscripts i, j). For the partitioning induced by p, let X^p be the union of solutions for the covering subproblems induced by every block B that were obtained by a ρ-approximation algorithm. Our algorithm then returns $X = \arg\min_p w(X^p)$ as the solution to the covering problem. Generalizing (and restating) the result of Huang et al. (phase 2 in the proof of Theorem 1 in [10]) we have that X is a $(\rho + O(1)/k)$-approximation.

Lemma 1 (Generalized formulation of [10]). *Solution X is a $(\rho + O(1)/k)$-approximation for the covering problem.*

In the following section we present a 4-approximation algorithm for the covering subproblem induced by a block B. This together with the preceding discussion and Lemma 1 yields the main theorem of this paper.

Theorem 1. *There is a $(4+\epsilon)$-approximation algorithm for MWDS in unit disk graphs.*

With the node-weighted Steiner tree algorithm by Zou et al. [17] that has approximation ratio smaller than 3.875, we obtain:

Theorem 2. *There is a 7.875-approximation algorithm for MWCDS in unit disk graphs.*

3 4-Approximation for the Covering Subproblem

In this section we present a 4-approximation algorithm for the covering subproblem induced by a block B: given a block B of $k \times k$ squares S_{ij}, compute a minimum-weight set of disks that covers all points $\mathcal{C}_B$.

Let OPT_B denote the set of disks in an optimal solution for the covering subproblem. In the following, we will often write that the algorithm "guesses" certain properties of OPT_B. By this we mean that the algorithm enumerates all possible choices for the guess (there will be a polynomial number of such choices) and computes a solution for each choice. If a choice leads to an infeasible solution, the algorithm does not consider that solution anymore. The algorithm keeps the best solution found and outputs it at the end. In the analysis of the algorithm, we concentrate on the solution X_B^{guess} for which the algorithm makes the right guesses about OPT_B. As the output of the algorithm is at least as good as this solution, it is enough to show that the approximation ratio of X_B^{guess} is 4.

First, the algorithm guesses for each of the $k \times k$ squares S_{ij} in block B whether there is a disk from OPT_B in S_{ij}, or not. If yes, the algorithm also guesses one such disk (clearly, there are no more than $(n+1)^{k^2}$ guesses). The guessed disks are then added to the set X_B^{guess} (empty in the beginning) that will form a solution to the covering problem.

Next, for every square S_{ij} containing an uncovered point, the algorithm guesses whether there is a point in S_{ij} that is covered in OPT_B only by disks from regions UM and LM. The regions UM and LM lie above and below S_{ij}, respectively, and between the vertical lines that contain the vertical parts of the boundary of S_{ij} (see Fig. 1). We call such a point a *middle-unique* point. If there is a middle-unique point in S_{ij}, the algorithm further guesses whether there is a middle-unique point that is covered by a disk from UM and, if yes, the algorithm also guesses the "leftmost" and the "rightmost" (with respect to the resulting sandglass lines, see below) such point p_l and p_r (p_l and p_r can be the same point), together with the corresponding disks d_{p_l} and d_{p_r} from UM. Similarly, the algorithm guesses whether there is a middle-unique point that is covered by a disk from LM, and if yes, the algorithm also guesses the leftmost and rightmost such point q_l and q_r, together with the corresponding disks d_{q_l} and d_{q_r} from LM.

The leftmost and rightmost middle-unique points p_l, p_r, q_l, q_r define a special region inside S_{ij}, called the *sandglass* of S_{ij} [10]: The union of the *upper sandglass* and the *lower sandglass.* The upper sandglass is a region inside S_{ij} determined by the line of slope -1 going through p_l and the line of slope 1 going through p_r as outlined in Fig. 1. The lower sandglass is, defined in a similar way, a region inside S_{ij} determined by the line of slope 1 going through q_l and the line of slope -1 going through q_r. Note that the sandglass region can be empty, if there is no middle-unique point in S_{ij}. Note also that the guessed disks $d_{p_l}, d_{p_r}, d_{q_l}, d_{q_r}$ partially cover the sandglass, but there may be a region that is not covered by these four disks. We define a *sandglass point* to be a point that lies in a sandglass but is not covered by $d_{p_l}, d_{p_r}, d_{q_l}, d_{q_r}$. Again, we add these four disks to the solution set X_B^{guess}.

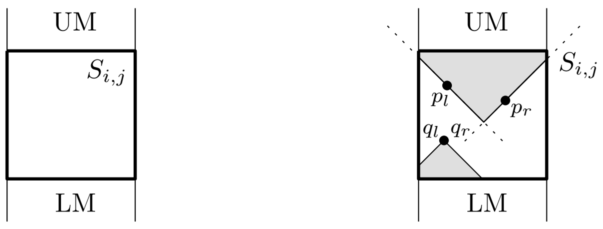

Fig. 1. Definition of regions UM and LM of a square S_{ij} (left figure). An example of a sandglass (the shaded area) for the case where $q_l = q_r$. For the upper sandglass, also the lines with slopes 1 and -1 are depicted (right figure)

The following lemma allows us to split the yet uncovered points in B into two parts, which we consider separately in the following. We say that a point *lies left of* S_{ij} if the x-coordinate of the point is smaller than the smallest x-coordinate of any point of S_{ij}. We define similarly in a natural way the notions of *lying right of*, *above*, and *below* S_{ij}.

Lemma 2 ([10]). *Any sandglass point is covered in* OPT_B *only by disks with center above or below* S_{ij}. *Any point of* S_{ij} *not contained in the sandglass of* S_{ij} *is covered in* OPT_B *only by disks with center left or right of* S_{ij}.

Using this lemma, we can partition the yet uncovered points into two parts. The *horizontal part* contains yet uncovered points that can be covered in OPT_B only by disks with center above or below the respective S_{ij}. The *vertical part* contains the rest of the yet uncovered points, i.e., the points that can be covered in OPT_B only by disks with center left or right of S_{ij}.

In the following we concentrate on the problem of covering the points in the horizontal part by a set of disks of minimum weight. We present a 2-approximation algorithm for this problem. Clearly, as the problem of covering the points in the vertical part can be solved by the same 2-approximation algorithm by rotating the setting by 90 degrees, we obtain a solution X_B^{guess} – all guessed disks plus the disks obtained by applying the 2-approximation algorithm to cover points in the horizontal and vertical part – which is a 4-approximation of OPT_B, thus showing the following theorem.

Theorem 3. *There is a 4-approximation algorithm for the covering subproblem in a block* B.

3.1 Covering Points Only From Above or Below

In the following we consider a generalized version of the covering problem of points in the horizontal part of block B. Let us consider k horizontal strips, each of height μ, containing m points P, where strip S_i, $i = 1, 2, \ldots, k$, lies between the horizontal lines $y = (i-1) \cdot \mu$ and $y = i \cdot \mu$. Let $\mathcal{D}$ be a set of n unit disks. We say that a disk d covers point $p \in P$ *from above*, if the center of disk d lies above the strip in which p is located. Similarly, we say that a disk d covers point $p \in P$ *from below*, if the center of disk d lies below the strip in which p is located. For a given set $X \subseteq \mathcal{D}$ of disks we say that *p is covered in X only from above or from below*, if p is covered by at least one disk from X, and for every disk $d \in X$, d covers p from below, or d covers p from above, or d does not cover p. We call the problem of covering points inside a horizontal strip of constant height with a minimum-weight set of disks for instances where there is an optimum solution such that every point $p \in P$ is covered in the optimal solution only from above or from below the *horizontal covering problem.*

Theorem 4. *There is a 2-approximation algorithm for the horizontal covering problem.*

Clearly, a constant-height horizontal strip can be seen as k strips of height μ, k being a constant. The simplest version of the problem is when $k = 1$. For this case, Ambühl et al. [1] present an algorithm that computes an optimum solution in polynomial time. The algorithm is based on the dynamic programming technique. The main idea is to consider the boundary of the disks that form an optimum solution inside the strip: the disks from above form in the strip the *upper envelope*, and the disks from below form in the strip the *lower envelope* of the optimum solution. The dynamic programming considers the points from left to right and stores, for each considered point and for each choice of current disks on the lower and upper envelope at that point, a minimum-weight set of disks that covers all points from the left up to the considered point.

The 2-approximation algorithm for the general case $k > 1$ can be obtained by extending this approach. Let $X \subseteq \mathcal{D}$ denote a feasible solution for the covering problem in k strips. In every strip S_i, $i = 1, \ldots, k$, we define the *upper envelope* U_i of X to be the intersection of the strip with the disks of X that lie above strip S_i. Similarly, the *lower envelope* L_i of X is the intersection of strip S_i with the disks of X that lie below S_i.

The algorithm uses a sweep line ℓ_i in every strip S_i to move on the boundary of the upper and lower envelope of every strip. Suppose we know an optimum solution OPT to the covering problem which covers every point only from above or below. Consider the upper and lower envelopes of OPT. We can sweep the lines ℓ_i through the solution OPT. All sweep lines l_i, $i = 1, 2, \ldots, k$, start somewhere to the left of the setting such that they do not intersect any disk or point. We move the sweep lines in discrete steps, always one line at a time. Every line l_i moves to the right, and visits (with its x-coordinate) the *corners* of the upper and lower envelope of strip S_i. A corner of an envelope is the intersection point of two disks which lies on the boundary of the envelope, or the intersection of a disk with one of the horizontal lines that delineate the strip, and the intersection lies on the boundary of the envelope. The sweeping process finishes when all corners of every strip have been visited. For this we make the sweep lines finish somewhere to the right of the setting, where no sweep line intersects a disk or a point of the setting. If we count the weight of every visited disk, at the end we end up with a weight that is at most three times $w(\mathrm{OPT})$, as every disk can be visited in at most three strips, and in every strip, we cannot count a disk more than once (as the disk cannot appear more than once on the boundary of an envelope [1]).

Our algorithm uses the sweeping approach to actually find a solution, i.e., to find the corners of envelopes which then define the disks in the final solution. We start with all sweep lines to the left of the setting. This indicates that no disk was chosen to cover a point in any strip. For every sweep line ℓ_i we remember the disks of the boundary of the upper and lower envelope that ℓ_i intersects at any time (for this we see the horizontal lines that define the strip as virtual disks of weight zero). The sweep line moves between corner points of the envelopes, so the disks we remember are the two disks that form the newly visited corner point, plus a disk that forms the boundary on the other envelope (upper or lower). We assume here that if a sweep line visits a corner of the upper (lower) envelope, then the lower (upper) envelope at this x-coordinate is formed by one disk only. We note that this assumption is without loss of generality, as we can initially rotate the whole problem setting so that this is true in every strip. For this purpose, we denote the current status of line ℓ_i by $((\bar{d}_l, \bar{d}_r), (\underline{d}_l, \underline{d}_r))$ with the meaning that $\bar{d}_l$ and $\bar{d}_r$ form the boundary of the upper envelope and $\underline{d}_l$ and $\underline{d}_r$ form the boundary of the lower envelope at the position of the sweep line l_i. Since we assume that at any position of the sweep line one of the two envelopes is formed by one disk only, we have $\bar{d}_l = \bar{d}_r$ or $\underline{d}_l = \underline{d}_r$. For our algorithm we require that a line ℓ_i can move from a corner point c to a corner point c' only when all points between c and c' are covered by the disks that form the boundary of the envelopes at c and at c'. (That is, for example, if line ℓ_i is at position x at state $((\bar{d}_l, \bar{d}_r), (\underline{d}_l, \underline{d}_r))$ and moves to position x' with state $((\bar{d}'_l, \bar{d}'_r), (\underline{d}'_l, \underline{d}'_r))$ then all points in strip S_i between x and x' have to be covered by disks $\bar{d}_l, \bar{d}_r, \underline{d}_l, \underline{d}_r, \bar{d}'_l, \bar{d}'_r, \underline{d}'_l, \underline{d}'_r$.) This restriction makes sure that if a sweep line gets from the start to the end, all points in the strip are covered by the chosen (visited) disks. If we do not pose any other restriction on the way the sweep line may move, we could use the dynamic programming approach of Ambühl et al. [1] for each strip individually and then combine the solutions of each strip to obtain a solution for the whole covering problem in k strips. As was shown in [10] this leads to a 3-approximation algorithm. The approximation ratio 3 comes from the fact that every disk can be counted three times, as it can appear as part of an upper or lower envelope in three strips.

We now show how to do sweeping in all strips simultaneously, achieving a better approximation ratio. We pose a new constraint on when a sweep line can move. Consider a disk d from strip S_i (i.e., the center of d lies in S_i). The disk can cover from above or from below points in at most three strips. Recall that the disk cannot cover any point in the strip S_i, as we are looking for solutions where every point is covered only from above or from below. Fig. 2 illustrates how a disk can intersect, besides S_i, two or three strips. In any case, a disk from strip S_i always intersects strips S_{i-1} and S_{i+1}. The constraint we pose on the sweep lines is that a line ℓ_{i-1} for which the lower envelope L_{i-1} is formed by disk d from strip S_i can move to the next corner point of L_{i-1} not formed by d

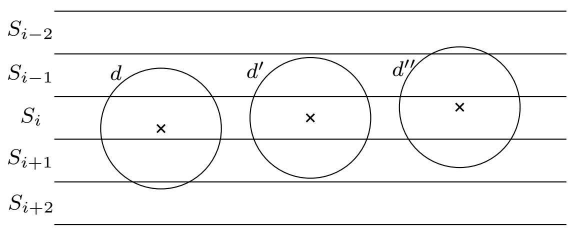

Fig. 2. Various examples of how a disk can cover points in strips. Disk d'' can cover points in S_{i-2} from below. Disks d, d' and d'' can cover points in S_{i-1} from below, and points in S_{i+1} from above. Disk d can cover points in S_{i+2} from below.

only if disk d has already appeared on the upper envelope of the sweep line ℓ_{i+1} in strip S_{i+1} (provided that it appears on that upper envelope at all). In other words, sweep line ℓ_{i-1} at L_{i-1} formed by d can move and "leave behind" disk d only if sweep line ℓ_{i+1} has already "met" d. If this is not the case, the line ℓ_{i-1} cannot move and we say that ℓ_{i-1} *waits* for the sweep line ℓ_{i+1}. Naturally, we pose a similar constraint for the line ℓ_{i+1} with respect to line ℓ_{i-1}, i.e., line ℓ_{i+1} can move from a corner point (d, d') of the upper envelope U_{i+1} to the right and "leave behind" disk d only if the sweep line ℓ_{i-1} has already "met" the disk d in strip S_{i-1}. We call these constraints the *move compatibility* constraints.

While sweeping through the strips, we count the weight of disks that were visited (i.e., the weight of disks that form the corner points which the sweep lines visit). We do not count, however, the weight of disk d every time (otherwise we would obtain a 3-approximation). If a line ℓ_i moves from a corner point (d, d') to a corner point (d', d''), the weight of disk d'' is added to the considered total weight only if at that moment no other sweep line contains the disk d'' already. Assume without loss of generality that d'' forms the lower envelope L_i in S_i. Then, with the previously posed constraint on how the sweep lines can move, we count the weight of the disk d'' in strips S_i and S_{i+2} only once. Thus, in total, the weight of any disk d'' used in the solution found by sweeping the lines in the strips is counted at most twice. This motivates the sweep lines to visit already used disks, as subsequent visits of a visited disk can cover points at no cost. This is the main reason why we get a 2-approximation algorithm.

We want to find a minimum-weight solution that moves the sweep lines from left to right and covers all points with visited disks. To find such a solution, we construct an auxiliary graph G_A and compute a shortest path in this graph. The vertex set V_A of the auxiliary graph G_A contains every possible configuration of the sweep lines. We will interchangeably call a vertex of G_A a configuration of the sweep lines. Clearly, every sweep line can be in at most n^3 different configurations, as there are at most n^2 corner points in every strip, and thus for any sweep line at a corner point, there can be at most n other disks forming the boundary of the other envelope. Thus, having k strips, there are no more than $O\left((n^3)^k\right)$ vertices in G_A. There are two special configurations. The *start vertex* (or the *start configuration*) s corresponds to the situation when all sweep lines are left of any disk, i.e., no disk forms an upper or lower envelope in any strip. Similarly, the *target vertex* t of G_A corresponds to the configuration where every sweep line is right of any disk. We connect the vertices in G_A with weighted edges. There is an edge between two configurations v and v' if one move of a sweep line ℓ_i in v results into the configuration v', and the move of the sweep line obeys the rule that all points between the original and new position of the moved sweep line are covered by the disks that the sweep line registers. Let d'' be the disk that forms the corner point in v' where the line ℓ_i moved to, but in v the disk was not part of the envelopes in S_i. The weight of the edge connecting v and v' is zero, if the disk d'' appears at another sweep line in v, otherwise the weight of the edge is $w_{d''}$, the weight of the disk d''.

Our algorithm finds a shortest path in G_A from s to t. This can be done in polynomial time if k is a constant. The computed path determines a move of the sweep lines from s to t, and the disks that the sweep lines meet is the solution of our algorithm. If there exists a path between s and t then clearly any such path gives a solution to the covering problem in k strips. In the following we show that the considered optimum solution OPT for the covering problem induces a path between s and t that additionally satisfies the move compatibility constraints. As our algorithm computes a shortest path between s and t, the total weight incurred by the sweep lines of our algorithm is at most the total weight incurred by the sweep lines that follow the s-t path induced by OPT. As we have argued above, the total weight of the s-t path induced by OPT is at most twice the weight of disks in OPT, as every disk in OPT can be counted by the sweep lines at most twice. Thus, the solution to the covering problem produced by our algorithm is at most twice the weight of OPT, which shows that the algorithm is a 2-approximation algorithm.

Lemma 3. *Let G_A be the auxiliary graph of the covering problem in k strips. Let* OPT *be an optimum solution for the problem. There is a path from s to t in G_A that corresponds to* OPT*, i.e., the disks visited on the s-t path are exactly the disks of* OPT*, and satisfies the move compatibility constraints.*

Proof. We will prove the claim by showing that at no point of time the sweep lines traversing the optimum solution OPT get stuck, i.e., we show that there is always a sweep line that can move to the right (unless, of course, the sweep lines are at the target configuration t).

Clearly, at the beginning, all sweep lines are left of any disk (the configuration s), and all sweep lines can move to the first disk in their respective strip (or to the end, if there is no disk of OPT in the strip). Assume for a contradiction that later in time, at a configuration $v \neq t$, no sweep line can move to the right, i.e., every sweep line ℓ_i that is not right of all disks waits for another sweep line to move first. We say that the lines are in a *deadlock*.

Let S_{i^*} be the strip with the minimum index i, $i = 1, 2, \ldots, k$, such that a sweep line ℓ_i waits for another sweep line to move. Thus, from the minimality of i^*, the sweep line ℓ_{i^*} waits for a sweep line ℓ_{i^*+2} to move. As we assume the sweep lines are in a deadlock, sweep line ℓ_{i^*+2} waits for another sweep line –

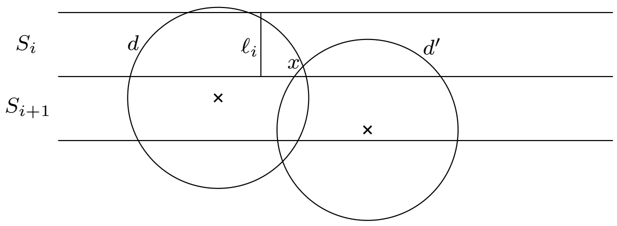

Fig. 3. Illustration for the proof of Lemma 3

ℓ_{i^*+2} waits either for ℓ_{i^*} or for ℓ_{i^*+4}. We will later show that no two sweep lines ℓ_i and ℓ_{i+2} can mutually wait for each other. Therefore, ℓ_{i^*+2} does not wait for ℓ_{i^*}, and it thus waits for ℓ_{i^*+4}. Then as the lines are in deadlock, ℓ_{i^*+4} waits either for ℓ_{i^*+2} or for ℓ_{i^*+6}. Using the same argument, ℓ_{i^*+4} waits for ℓ_{i^*+6}. Thus, using this argumentation iteratively, we end up claiming that ℓ_{i^*+2j} waits for $\ell_{i^*+2(j+1)}$, for any $j \geq 0$. This is not possible, as there are only k sweep lines.

We are left to show that the situation in which sweep lines ℓ_i and ℓ_{i+2} wait for each other does not occur. Assume such a situation. Sweep line ℓ_i waits for sweep line ℓ_{i+2} because ℓ_i wants to leave a disk d but the line ℓ_{i+2} did not pass the disk d in strip S_{i+2} yet. Similarly, line ℓ_{i+2} wants to leave a disk d' but the line ℓ_i did not pass the disk d' in strip S_i. We show that these assumptions give contradicting claims on the position of the disks d and d'. Consider now the disks d and d' alone, i.e., without the other disks of OPT. Now, as ℓ_i is currently at disk d and the line did not pass the disk d' yet, disk d' has to appear in S_i after d. This implies, however, that the center of d' is strictly right of the center of d (in terms of the x-coordinates). Fig. 3 illustrates this situation. Observe first that if disk d' (which appears right of d in S_i) intersects disk d, say at point x, then disk d' can be seen as a rotation of disk d around point x in counterclockwise direction. As the rotation leaves the center of the disk in the strip below, the rotation translates the center of the disk strictly to the right. If the disk d' does not intersect d, we can move the disk d to the right until the first moment when the translated disk d intersects d'. Repeating the argument we see that the center of d' is strictly right of the center of d.

Similarly, we can argue for the positions of disks in strip S_{i+2}, leading to the claim that the center of d' is strictly left of the center of d. This is a contradiction and the lemma follows. □

4 Conclusions

In this work we have presented a $(4+\epsilon)$-approximation algorithm for the problem of computing a minimum-weight dominating set in unit disk graphs. The main ingredient is a new 4-approximation algorithm for settings restricted to constant-size squares. This, in turn, uses a new 2-approximation algorithm for the problem of covering points in a constant-height strip only by disks from above or below. The 2-approximation algorithm finds a solution by computing a shortest path in an auxiliary graph that can be seen as mimicking a sweep-line approach with k sweep lines, which in turn mimic computing k parallel dynamic programs. This technique may be of independent interest. It remains open whether MWDS in unit disk graphs admits a PTAS.

References

1. C. Ambühl, T. Erlebach, M. Mihalák, and M. Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In *Proc. 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 3–14, 2006.
2. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.
3. X. Cheng, X. Huang, D. Li, W. Wu, and D.-Z. Du. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003.
4. B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3):165–177, 1990.
5. D. Dai and C. Yu. A $5 + \epsilon$-approximation algorithm for minimum weighted dominating set in unit disk graph. *Theoret. Comput. Sci.*, 410(8-10):756–765, 2009.
6. U. Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.
7. M. R. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, San Francisco, 1979.
8. S. Guha and S. Khuller. Improved methods for approximating node weighted Steiner trees and connected dominating sets. *Inform. and Comput.*, 150(1):57–74, 1999.
9. D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.
10. Y. Huang, X. Gao, Z. Zhang, and W. Wu. A better constant-factor approximation for weighted dominating set in unit disk graph. *J. Comb. Optim.*, 2008.
11. H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998.
12. D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.
13. M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995.
14. G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 770–779, 2000.
15. E. J. van Leeuwen. Approximation algorithms for unit disk graphs. In *Proc. 31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 351–361, 2005.
16. V. V. Vazirani. *Approximation Algorithms.* Springer, 2001.
17. F. Zou, X. Li, S. Gao, and W. Wu. Node-weighted steiner tree approximation in unit disk graphs. *Theoret. Comput. Sci.*, 18(4):342–349, 2009.
18. F. Zou, Y. Wang, X.-H. Xu, X. Li, H. Du, P. Wan, and W. Wu. New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs. *Theoret. Comput. Sci.*, 2009. Article in Press.