SIAM J. Comput. © 2005 Society for Industrial and Applied Mathematics
Vol. 34, No. 6, pp. 1302–1323

POLYNOMIAL-TIME APPROXIMATION SCHEMES FOR GEOMETRIC INTERSECTION GRAPHS[*]

THOMAS ERLEBACH[†], KLAUS JANSEN[‡], AND EIKE SEIDEL[‡]

Abstract. A disk graph is the intersection graph of a set of disks with arbitrary diameters in the plane. For the case that the disk representation is given, we present polynomial-time approximation schemes (PTASs) for the maximum weight independent set problem (selecting disjoint disks of maximum total weight) and for the minimum weight vertex cover problem in disk graphs. These are the first known PTASs for $\mathcal{NP}$-hard optimization problems on disk graphs. They are based on a novel recursive subdivision of the plane that allows applying a shifting strategy on different levels simultaneously, so that a dynamic programming approach becomes feasible. The PTASs for disk graphs represent a common generalization of previous results for planar graphs and unit disk graphs. They can be extended to intersection graphs of other "disk-like" geometric objects (such as squares or regular polygons), also in higher dimensions.

Key words. independent set, vertex cover, shifting strategy, disk graph

AMS subject classifications. 68Q25, 68R10

DOI. 10.1137/S0097539702402676

1. Introduction.

Intersection graphs are graphs whose vertices are represented by sets such that two vertices are adjacent if and only if the corresponding sets have a nonempty intersection. They have been studied by many authors [12, 6, 24]. We are interested in approximation algorithms for $\mathcal{NP}$-hard optimization problems on intersection graphs of geometric objects, in particular approximation algorithms for independent set and vertex cover. Two prominent applications of geometric intersection graphs are frequency assignment in cellular networks [13, 22] and map labeling [1].

The goal of the *maximum weight independent set* problem (MWIS) is to compute, for a given set of geometric objects with certain weights, a subset of disjoint (nonoverlapping) objects with maximum total weight. The goal of the *minimum weight vertex cover* problem (MWVC) is to compute a subset of the given objects with minimum total weight such that, for any two intersecting objects, at least one of the objects is contained in the subset. MIS and MVC refer to the unweighted versions of these problems. We obtain polynomial-time approximation schemes for MWIS and MWVC in the intersection graphs of disks, squares, or other "disk-like" objects, also in higher dimensions.

1.1. Preliminaries.

For a set V of geometric objects, the corresponding *intersection graph* is the undirected graph with vertex set V and an edge between two

[*]Received by the editors February 15, 2002; accepted for publication (in revised form) August 2, 2004; published electronically June 27, 2005. A preliminary version of this paper was presented at the 12th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2001). This research was supported in part by DFG Graduiertenkolleg 357, Effiziente Algorithmen und Mehrskalenmethoden, by EU Research Training Network ARACNE, Approximation and Randomized Algorithms in Communication Networks, HPRN-CT-2000-00112, and by EU Thematic Network APPOL I + II, Approximation and Online Algorithms, IST-1999-14084 and IST-2001-32007.

http://www.siam.org/journals/sicomp/34-6/40267.html

[†]Department of Computer Science, University of Leicester, University Road, Leicester LE1 7RH, UK (t.erlebach@mcs.le.ac.uk).

[‡]Institute for Computer Science and Applied Mathematics, Christian-Albrechts-University of Kiel, Olshausenstr. 40, D-24098 Kiel, Germany (kj@informatik.uni-kiel.de, ese@informatik.uni-kiel.de).

vertices if the corresponding objects intersect.

Assume that we are given a set $\mathcal{D} = \{D_1, \ldots, D_n\}$ of n (topologically closed) disks in the plane, where D_i has diameter d_i, center $c_i = (x_i, y_i)$, and weight w_i. For a subset $U \subseteq \mathcal{D}$, $w(U)$ denotes the sum of the weights of the disks in U. Disks D_i and D_j *intersect* if $dist(c_i, c_j) \leq (d_i + d_j)/2$, where $dist(p_1, p_2)$ denotes the Euclidean distance between two points p_1 and p_2 in the plane. A *disk graph* is the intersection graph of a set of disks. We assume that the input to our algorithms is the set $\mathcal{D}$ of disks, not only the corresponding intersection graph. This is an important distinction, because determining for a given graph whether it is a disk graph is known to be $\mathcal{NP}$-hard [16], and hence no efficient method is known for computing a disk representation if only the intersection graph is given.

Interestingly, every planar graph is a *coin graph*, i.e., the intersection graph of a set of interior-disjoint disks [21]. Therefore, the class of disk graphs properly contains the class of planar graphs.

For a given set $\mathcal{D}$ of disks in the plane, we let $OPT_{IS}(\mathcal{D})$ and $OPT_{VC}(\mathcal{D})$ denote the total weight of an optimal solution for MWIS and MWVC, respectively. An algorithm is a ρ-approximation algorithm for MWIS if it runs in polynomial time and always computes an independent set of total weight at least $\frac{1}{\rho} OPT_{IS}(\mathcal{D})$. An algorithm is a ρ-approximation algorithm for MWVC if it runs in polynomial time and always computes a vertex cover of total weight at most $\rho OPT_{VC}(\mathcal{D})$. An algorithm is a *polynomial-time approximation scheme* (PTAS) for MWIS if it takes an additional parameter $\varepsilon > 0$ and always computes an independent set of total weight at least $\frac{1}{1+\varepsilon} OPT_{IS}(\mathcal{D})$, where the running-time is polynomial in the size of the representation of $\mathcal{D}$ for fixed $\varepsilon > 0$. A PTAS for MWVC is defined analogously. If an algorithm is a ρ-approximation algorithm, the algorithm is also said to have approximation ratio ρ.

Note that the complement of an independent set is a vertex cover, and vice versa. Therefore, we have $OPT_{IS}(\mathcal{D}) = w(\mathcal{D}) - OPT_{VC}(\mathcal{D})$. Nevertheless, taking the complement of the solution output by a ρ-approximation algorithm for MWIS does in general not provide a ρ-approximation for MWVC, and vice versa.

In general graphs with n vertices, there cannot be a polynomial-time approximation algorithm for MWIS with approximation ratio $n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless $\mathcal{NP} = \text{co-}RP$ [14]. MWVC is MAX SNP-hard in general graphs and cannot be approximated within a constant smaller than 7/6 unless $\mathcal{P} = \mathcal{NP}$ [15]. For MWVC in general graphs, a 2-approximation algorithm is known [3]. For intersection graphs of geometric objects, better approximation ratios are often possible.

1.2. Related work on disk graphs. For unit disk graphs (intersection graphs of disks with equal diameter), MWIS and MWVC remain $\mathcal{NP}$-hard [9], but PTASs exist for MWIS, MWVC, and the minimum dominating set problem if the disk representation is given as part of the input [18]. For intersection graphs of disks with arbitrary diameters, the best previously known approximation algorithms achieve approximation ratio 5 for MIS [23] and $\frac{3}{2}$ for MWVC [22]. In [18] and [22], the question was raised whether a PTAS exists for disk graphs. As the class of disk graphs contains the class of unit disk graphs and the class of planar graphs, a PTAS for disk graphs would generalize the results for unit disk graphs due to Hunt et al. [18] and the results for planar graphs due to Baker [2]. This paper resolves this question by presenting PTASs for MWIS and MWVC in disk graphs (with given representation).

A PTAS for the fractional chromatic number problem on disk graphs, using our PTAS for MWIS as a subroutine, was obtained in [20, 19].

1.3. Related work on map labeling. Map labeling refers to a family of tasks concerning the placement of labels on a map. Often it is assumed that the features to be labeled are points and that the labels can be modeled as rectangles (just take the bounding box of the label text). For each feature, there are certain admissible positions of the labeling rectangle: for example, the rectangle must be placed so that the feature coincides with a corner of the rectangle. Rectangles can be assigned weights that represent the importance of including that label on the map. Then it is meaningful to study the problem of maximizing the total weight of labeled features subject to the constraint that different labels must not overlap. This is just MWIS in the intersection graph of a set of (topologically closed) axis-aligned rectangles.

Agarwal, van Kreveld, and Suri [1] give an $O(\log n)$-approximation algorithm for MIS in an intersection graph of n axis-aligned rectangles. Their algorithm can be adapted to the weighted case in a straightforward way, thus yielding an approximation ratio of $O(\log n)$ also for MWIS. For the special case that all rectangles have the same height (which is meaningful if the labels are text labels of a certain font size), they obtain a PTAS.

Doddi et al. [10] consider *sliding labels* (a point can lie anywhere on the boundary of its label) and assume that labels may be placed in any orientation. They provide constant-factor approximation algorithms for maximizing the size of the labels (assuming that all features must be labeled and that all labels are circles or squares with identical size). These were recently improved in [11]. In [10], bicriteria approximation algorithms that label a $(1-\varepsilon)$-fraction of all features with labels whose size is at least a $\frac{1}{1+\varepsilon}$-fraction of the optimal size are also discussed. These algorithms use a PTAS for MWIS as a subroutine. In [10], it is mentioned that their algorithms can be extended to the case of nonuniform squares if the ratio between the size of the largest square and the smallest square is bounded by a constant. Using our new PTAS for MWIS in the intersection graphs of squares, their algorithms can now be extended to labeling with nonuniform squares where this ratio is arbitrary.

Van Kreveld, Strijk, and Wolff [27] investigate the question of how many features in a map can be labeled with rectangular labels if different restrictions on the labeling model are enforced (feature must be at a corner of the rectangle versus sliding rectangles). They present a practical 2-approximation algorithm and a PTAS for maximizing the number of labeled features in the case of sliding rectangular labels of equal height.

An up-to-date bibliography of publications on map labeling can be found on the Web [28].

1.4. Our results.

In this paper we present PTASs for MWIS and MWVC in the intersection graphs of “disk-like” objects. In sections 2 and 3, we present the details of the PTASs for MWIS and MWVC in disk graphs. In section 4, we discuss how our approach can be extended to other geometric objects (such as squares or regular polygons) and to higher dimensions. We give our conclusions and mention some open problems in section 5.

Our PTASs are based on a sophisticated use of the *shifting strategy* [2, 17] that was previously employed, among other results, for obtaining PTASs for various optimization problems in planar graphs [2] and unit disk graphs [18]. We partition the given disks into levels according to their diameters and use a novel recursive subdivision of the plane that allows us to apply the shifting strategy on all levels simultaneously.

We outline the basic idea of the PTAS for MWIS. The plane is partitioned into squares on each level, and some of the disks are removed from the input so that different squares on the same level yield independent subproblems with respect to all disks that are on this level or on a level with disks of smaller diameter. Furthermore, at most a constant number of disks with larger diameter can be disjoint and intersect a square on the current level. Hence, all such sets of disks can be enumerated in polynomial time for each square, and a dynamic programming approach becomes feasible. The details are given in the next section.

2. A PTAS for independent set in disk graphs.

2. A PTAS for independent set in disk graphs. Let $k > 1$ be a fixed positive integer. Scale all disks so that the largest disk has diameter 1. Let $d_{\min}$ be the diameter of the smallest disk. Let $\ell = \lfloor \log_{k+1}(1/d_{\min}) \rfloor$. We partition the given set $\mathcal{D}$ of disks into $\ell + 1$ levels. For $0 \le j \le \ell$, level j consists of all disks D_i with diameter d_i satisfying $(k+1)^{-j} \ge d_i > (k+1)^{-(j+1)}$. Note that the disk with diameter $d_{\min}$ is on level ℓ.

An example with three levels is sketched in Figure 2.1.

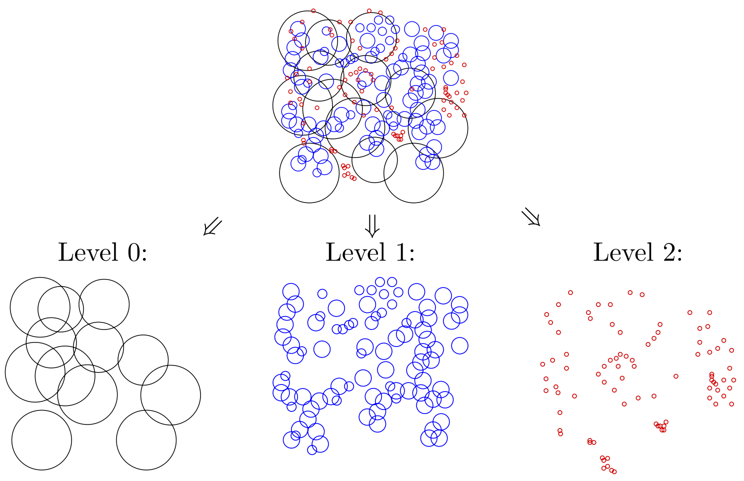

FIG. 2.1. *Partitioning the disks into levels* ($k = 2$).

2.1. Subdividing the plane.

2.1. Subdividing the plane. For each level j, $0 \le j \le \ell$, we impose a grid on the plane that consists of lines that are $(k+1)^{-j}$ apart from each other. The vth vertical line, $v \in \mathbb{Z}$, is at $x = v(k+1)^{-j}$. The hth horizontal line, $h \in \mathbb{Z}$, is at $y = h(k+1)^{-j}$. We say that the vth vertical line has index v and that the hth horizontal line has index h. Furthermore, we say that a disk D_i with center (x_i, y_i) and diameter d_i *hits* a vertical line at $x = a$ if $a - d_i/2 < x_i \le a + d_i/2$. Similarly, we say that D_i hits a horizontal line at $y = b$ if $b - d_i/2 < y_i \le b + d_i/2$. Intuitively, a disk hits a line if it intersects that line, except if it only touches the line from the left or from below. Note that every disk can hit at most one horizontal line and at most one vertical line on its level.

Let $0 \le r, s < k$ and consider the vertical lines whose index modulo k equals r and the horizontal lines whose index modulo k equals s. We say that these lines are *active* for (r, s). Figure 2.2 illustrates the horizontal grid lines and active lines on two consecutive levels.

Define $\mathcal{D}(r, s)$ to be the set of disks that is obtained from $\mathcal{D}$ by deleting all disks that hit a line that is on the same level as the disk and that is active for (r, s). See Figure 2.3 for an example.

In the following, we write OPT as shorthand for OPT_{IS}.

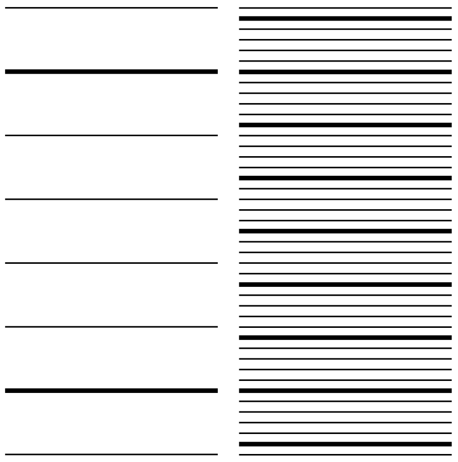

FIG. 2.2. *Horizontal grid lines on level j (left-hand side) and level $j+1$ (right-hand side) for $k = 5$. Active lines are drawn bold.*

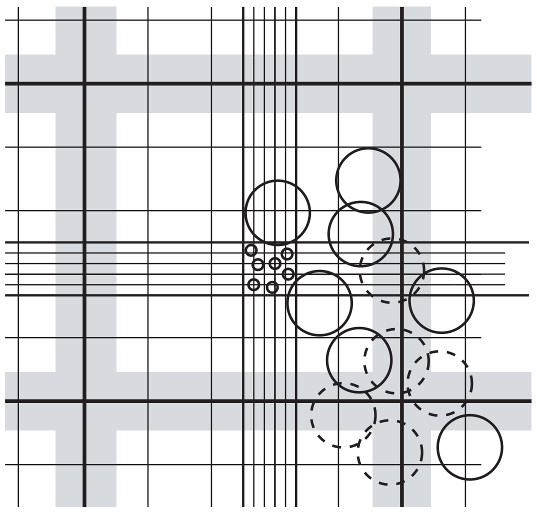

FIG. 2.3. *Example of grid and active lines on level j (coarse grid) and on level $j+1$ (fine grid) for $k = 5$. The big disks have level j, and the small disks have level $j+1$. All disks shown have the maximum possible diameter on their level. Active lines are drawn bold. Disks that hit active lines are drawn dashed. Note that a disk on level j can hit an active line only if its center is in the shaded strip along that active line.*

LEMMA 2.1. *For at least one pair (r, s), $0 \leq r, s < k$, we have $OPT(\mathcal{D}(r, s)) \geq (1 - \frac{1}{k})^2 OPT(\mathcal{D})$.*

Proof. Let $S^* \subseteq \mathcal{D}$ be any set of disjoint disks with total weight $OPT(\mathcal{D})$.

For $0 \leq r < k$, let S_r^* be the set of all disks in S^* that hit a vertical line on their level whose index modulo k is r. As the sets S_r^* are disjoint, the weight of at least one of them must be at most a $\frac{1}{k}$-fraction of the weight of S^*. For this set S_r^*, let $T^* = S^* \setminus S_r^*$ and note that the weight of T^* is at least $(1 - \frac{1}{k})OPT(\mathcal{D})$.

For $0 \leq s < k$, let T_s^* be the set of all disks in T^* that hit a horizontal line on their level whose index modulo k is s. The weight of at least one of these sets T_s^*

must be at most a $\frac{1}{k}$-fraction of the weight of T^*. For this set T^*_s, let $U^* = T^* \setminus T^*_s$. Note that $U^* \subseteq \mathcal{D}(r,s)$ and the weight of U^* is at least $(1-\frac{1}{k})^2 OPT(\mathcal{D})$. ∎

The algorithm considers all k^2 possible values for r and s such that $0 \le r, s < k$. For each possibility, an optimal independent set in $\mathcal{D}(r,s)$ is computed using dynamic programming. Among the k^2 sets obtained in this way, the one with largest weight is output. By Lemma 2.1, this set has total weight at least $(1-\frac{1}{k})^2 OPT(\mathcal{D})$. Therefore, the algorithm achieves approximation ratio $(1+\frac{1}{k-1})^2$. As k gets larger, the approximation ratio gets arbitrarily close to 1.

It remains to show how an optimal independent set in $\mathcal{D}(r,s)$ can be computed using dynamic programming and that the running-time of the algorithm is polynomial in the size of the input for a fixed value of k.

2.2. Dynamic programming. Let $0 \le r, s < k$. In this section we discuss the dynamic programming algorithm for computing an optimal independent set in $\mathcal{D}(r,s)$.

Consider one particular level j, $0 \le j \le \ell$. The lines on level j that are active for (r,s) partition the plane into squares. More precisely, for consecutive active vertical lines at $x = a_1$ and $x = a_2$ and consecutive active horizontal lines at $y = b_1$ and $y = b_2$, one square $\{(x,y) \mid a_1 < x \le a_2, b_1 < y \le b_2\}$ is obtained. We refer to these squares on level j as *j-squares.*

By definition of $\mathcal{D}(r,s)$, every disk in $\mathcal{D}(r,s)$ that is on level j is completely contained in some j-square. Furthermore, we have the following lemma about the relationship between squares on different levels.

LEMMA 2.2. *For any j, $0 \le j < \ell$, every $(j+1)$-square is completely contained in some j-square.*

Proof. We prove the lemma by showing that every line that is active for (r,s) on level j is also active for (r,s) on level $j+1$. Figure 2.2 illustrates this claim for the horizontal lines: note that the active lines on level j are active on level $j+1$ also after the active lines are "shifted" up or down on their respective levels.

Without loss of generality, consider only the horizontal lines. Let $y = h(k+1)^{-j}$ be an active horizontal line on level j. This means that $h \bmod k = s$. This line is identical to the line $y = h(k+1)(k+1)^{-(j+1)}$, which is on level $j+1$ and has index $h(k+1)$. Obviously, $h(k+1) \bmod k = h \bmod k = s$. Hence, the line is also active on level $j+1$. ∎

COROLLARY 2.3. *Every j-square is the union of $(k+1)^2$ $(j+1)$-squares.*

Call a j-square S *relevant* if $\mathcal{D}(r,s)$ contains at least one disk of level j that is contained in S. For a relevant j-square S and a relevant j'-square S', $j' > j$, we say that S' is a *child* or *child square* of S (and S is a *parent* of S') if S' is contained in S and if there is no relevant j''-square S'', $j' > j'' > j$, such that S' is contained in S'' and S'' is contained in S.

The algorithm processes all relevant squares in order of nonincreasing levels. When a j-square S is processed, a table T_S is computed. For every set I of disjoint disks of level smaller than j that intersect S, the table entry $T_S(I)$ is a maximum weight set of disjoint disks of level at least j that are contained in S and disjoint from the disks in I. To formalize this property, we introduce the following definition.

DEFINITION 2.4. *Let S be a relevant j-square and let I be a set of disjoint disks of level less than j that intersect S. Then the table entry $T_S(I)$ is called* good *if it satisfies the following properties:*

(a) *$T_S(I) \subseteq \mathcal{D}(r,s)$ consists of disks that are contained in S and have level at least j.*

(b) *$I \cup T_S(I)$ is an independent set.*

(c) *$w(T_S(I))$ is maximum among all sets that satisfy* (a) *and* (b).

Provided that tables with good entries have been computed for all relevant squares, it is clear that the algorithm can output a maximum weight independent set in $\mathcal{D}(r,s)$ by taking the union of the sets $T_S(\emptyset)$ for all relevant squares S that do not have a parent. In the next section, we give an algorithm to efficiently compute the table T_S for a j-square S provided that the tables $T_{S'}$ have already been computed for all child squares S' of S.

2.3. Computing the table for a relevant square. Consider some relevant j-square S. First, we give a bound on the number of sets I for which a table entry $T_S(I)$ needs to be computed. For this purpose, we show that the number of disjoint disks of level smaller than j that can intersect a j-square is bounded by $O(k^2)$. Figure 2.3 can serve as an illustration of this fact: only $O(k^2)$ disjoint disks whose diameter is larger than that of the big disks can intersect the j-square shown in the figure.

LEMMA 2.5. *Let S be some j-square and let $I \subseteq \mathcal{D}$ be a set of disjoint disks such that each disk in I has level at most $j-1$ and intersects S. Then there is a constant C such that $|I| \leq Ck^2$.*

Proof. Let $\bar{S}$ be a square that consists of S and a strip of width $(k+1)^{-j}$ surrounding S. As the disks in I have level at most $j-1$, they have diameter larger than $(k+1)^{-j}$ and, therefore, area larger than $\pi((k+1)^{-j}/2)^2$. Furthermore, each disk in I occupies an area larger than $\pi((k+1)^{-j}/2)^2$ within $\bar{S}$: a disk of diameter $(k+1)^{-j}$ that intersects S would have to be completely contained in $\bar{S}$, and larger disks that intersect S would have to occupy an area of $\bar{S}$ that is even larger. The area of $\bar{S}$ is $((k+2)(k+1)^{-j})^2$. Therefore, we must have

$$|I| \leq \frac{((k+2)(k+1)^{-j})^2}{\pi((k+1)^{-j}/2)^2} = \frac{4}{\pi}(k+2)^2 < 6k^2.$$

Hence, we can choose $C = 6$. ∎

By Lemma 2.5, the algorithm can enumerate all independent sets I of disks that have level smaller than j and that intersect S as follows: It simply enumerates all subsets of at most Ck^2 disks of level smaller than j that intersect S and checks for each of them whether it is an independent set. This enumeration can be performed in time $n^{O(k^2)}$.

We consider one such set I and show how $T_S(I)$ is computed. Assume for now that the j-square S has either no child square at all or exactly $(k+1)^2$ child squares on level $j+1$. Denote the child squares by $S'_{g,h}$, where g is the row index and h is the column index, $0 \leq g, h \leq k$. Thus, $S'_{0,0}$ is the bottom left child square of S, and $S'_{k,k}$ is the top right child square. We will show later how to deal with the case that some child squares of S are at a level larger than $j+1$.

We can assume that $T_{S'_{g,h}}$ has already been computed for each such child square $S'_{g,h}$. A first approach to computing $T_S(I)$, which we have used in an earlier version of our result, is to enumerate all sets I' of disjoint disks of level j that intersect S and that are disjoint from the disks in I, and to look up, for each such set I', the table entries $T_{S'_{g,h}}(I'_{g,h})$, where $I'_{g,h}$ is the set of disks in $I \cup I'$ that intersect $S'_{g,h}$. The union of I' and all sets of the form $T_{S'_{g,h}}(I'_{g,h})$ forms a candidate set, and the candidate set of largest weight yields $T_S(I)$. The cardinality of I' can be bounded by $O(k^4)$, and so this approach leads to a running-time of $n^{O(k^4)}$. In the following, we present an improved algorithm based on dynamic programming that allows us to reduce the running-time to $n^{O(k^2)}$.

By $S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$, where $0 \le g_1 \le g_2 \le k$ and $0 \le h_1 \le h_2 \le k$, we denote the union of all child squares $S'_{g,h}$ with $g_1 \le g \le g_2$ and $h_1 \le h \le h_2$. We call such a union of child squares a *rectangle*. We say that a disk D *intersects the boundary* of a rectangle R if $D \cap R$ and $D \setminus R$ are both nonempty.

In order to determine $T_S(I)$, the algorithm computes an auxiliary table $AT_{S,I}$ with entries $AT_{S,I}(S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}, J)$ for certain values of g_1, g_2, h_1, h_2, where J is a set of disks of level j that intersect the boundary of $S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$ and have the property that $I \cup J$ is an independent set. The table entry $AT_{S,I}(S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}, J)$ is a maximum weight set J' of disks that have level at least j, are contained in $S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$, and have the property that $I \cup J \cup J'$ is an independent set. This property is captured formally in the following definition.

Definition 2.6. *Let S be a relevant j-square and let I be a set of disjoint disks of level less than j that intersect S. Let $S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$ be a rectangle of child squares of S, and let J be a set of disks of level j that intersect the boundary of $S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$ and have the property that $I \cup J$ is an independent set.*

Then the table entry $AT_{S,I}(S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}, J)$ is called good *if it satisfies the following properties:*

(a) *$AT_{S,I}(S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}, J) \subseteq \mathcal{D}(r,s)$ consists of disks that are contained in $S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$ and have level at least j.*

(b) *$I \cup J \cup AT_{S,I}(S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}, J)$ is an independent set.*

(c) *$w(AT_{S,I}(S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}, J))$ is maximum among all sets that satisfy* (a) *and* (b).

Once a good table entry $AT_{S,I}(S'_{0\cdot\cdot k,0\cdot\cdot k}, \emptyset)$ is computed, it immediately yields $T_S(I)$.

Table entries $AT_{S,I}(S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}, J)$ are again computed by dynamic programming. First, we bound the number of different sets J that must be considered as table index for a rectangle $S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$.

Lemma 2.7. *Let $S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$ be a rectangle and let J be a set of disjoint disks of level j such that all disks in J intersect the boundary of $S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$. Then there is a constant C' such that $|J| \le C'k^2$.*

Proof. Let $R = S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$. The boundary B of R consists of line segments with total length at most $4k(k+1)^{-j}$. Let $\bar{B}$ be obtained by extending B by a strip of width $(k+1)^{-(j+1)}$ on the inside of B and on the outside of B. The total area of $\bar{B}$ is at most $8k(k+1)^{-2j-1}$. Each disk of level j that intersects B occupies an area at least $\pi(k+1)^{-2(j+1)}/4$ of $\bar{B}$. Therefore, J can contain at most

$$\frac{8k(k+1)^{-2j-1}}{\frac{\pi}{4}(k+1)^{-2(j+1)}} = \frac{32}{\pi}k(k+1) \le 16k^2$$

disjoint disks. So we can take $C' = 16$. ∎

Lemma 2.7 shows that for each rectangle $S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}$, there are at most $n^{O(k^2)}$ different sets J that need to be considered as table index with respect to $AT_{S,I}$ $(S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}, J)$.

Here and in the following, we use $AT_{S,I}(S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}, *)$ as a shorthand notation to refer to the table entries $AT_{S,I}(S'_{g_1\cdot\cdot g_2,h_1\cdot\cdot h_2}, J)$ for all relevant values of J.

As the basis of the dynamic programming, the table entries $AT_{S,I}(S'_{g\cdot\cdot g,h\cdot\cdot h}, *)$ are computed for $0 \le g, h \le k$ as shown in Figure 2.4. Note that $S'_{g\cdot\cdot g,h\cdot\cdot h} = S'_{g,h}$. For each child square $S'_{g,h}$, all independent sets U of disks of level j intersecting $S'_{g,h}$ are enumerated. This can be done in time $n^{O(k^2)}$ by Lemma 2.5. If $I \cup U$ is

```
Input: square $S$ on level $j$,
       set $I$ of disjoint disks of level $< j$ intersecting $S$,
       integers $g, h$ with $0 \le g, h \le k$
Output: table entries $AT_{S,I}(S'_{g,h}, J)$ for all $J$
$AT_{S,I}(S'_{g,h}, *) \leftarrow$ undefined;
$Q \leftarrow$ all disks in $\mathcal{D}(r,s)$ of level $j$ intersecting $S'_{g,h}$;
for all $U \subseteq Q$ such that $|U| \le Ck^2$ do
    if the disks in $I \cup U$ are disjoint then
        $I' \leftarrow \{D \in I \mid D$ intersects $S'_{g,h}\}$;
        $X \leftarrow T_{S'_{g,h}}(I' \cup U)$;
        $X \leftarrow X \cup \{D \in U \mid D$ is contained in $S'_{g,h}\}$;
        $J \leftarrow \{D \in U \mid D$ intersects the boundary of $S'_{g,h}\}$;
        if $AT_{S,I}(S'_{g,h}, J)$ is undefined or
          $w(X) > w(AT_{S,I}(S'_{g,h}, J))$ then
            $AT_{S,I}(S'_{g,h}, J) \leftarrow X$;
        fi
    fi
od
```

FIG. 2.4. *Computing the auxiliary table $AT_{S,I}(S'_{g,h}, *)$.*

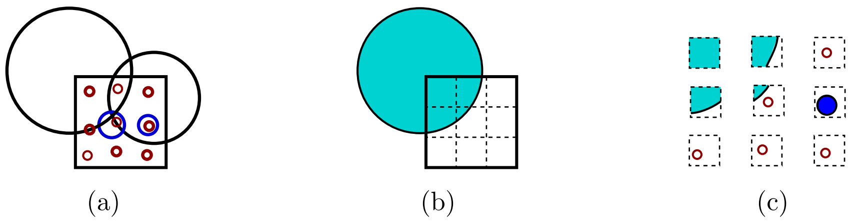

FIG. 2.5. *Example of table lookups for a square S at level j in case $k = 2$.* (a) *shows* 13 *disks in $\mathcal{D}(r,s)$ that intersect S:* 2 *disks of level less than j,* 2 *disks on level j, and* 9 *disks on level $j+1$.* (b) *displays an independent set I consisting of* 1 *disk of level less than j.* (c) *illustrates that lookups are performed in* 9 *tables $T_{S'_{g,h}}$ during the computation of the table entries $AT_{S,I}(S'_{g,h}, *)$.*

an independent set, the optimal way of extending the set $I \cup U$ to a larger weight independent set by adding disks of larger levels that are contained in $S'_{g,h}$ is computed by looking up $T_{S'_{g,h}}(I' \cup U)$, where $I' = \{D \in I \mid D \text{ intersects } S'_{g,h}\}$. (If S does not have any relevant child squares, all lookups in tables $T_{S'_{g,h}}$ are taken to return the empty set.) Let $J = \{D \in U \mid D \text{ intersects the boundary of } S'_{g,h}\}$. If the independent set obtained in this way has larger weight than the previous set stored in table entry $AT_{S,I}(S'_{g,h}, J)$, the entry is updated to store the new set. An example of the table lookups performed for a relevant j-square S in the tables $T_{S'_{g,h}}$ of subsquares at level $j+1$ is sketched in Figure 2.5.

Next, we show how to combine the information from the table entries of two rectangles to obtain the table entries for the rectangle representing the union of the two rectangles. Without loss of generality, we discuss only the case where the two rectangles share a horizontal edge and have the same width. The combination of rectangles that share a vertical edge and have the same height is analogous. It is clear that $(k+1)^2 - 1$ such combinations suffice to obtain the table entry $AT_{S,I}(S'_{0\cdot\cdot k,0\cdot\cdot k}, \emptyset)$,

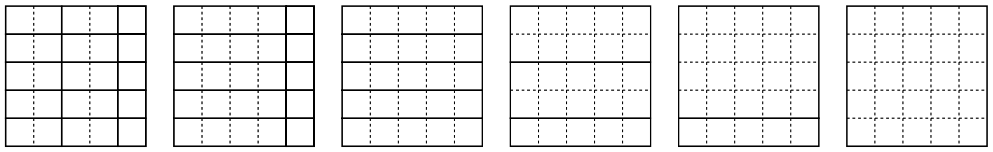

FIG. 2.6. *Combining subsquares into rectangles.*

```
Input: square $S$ on level $j$,
        set $I$ of disjoint disks of level $< j$ intersecting $S$,
        integers $g_1, g_2, g_3$ with $0 \leq g_1 \leq g_2 < g_3 \leq k$,
        integers $h_1, h_2$ with $0 \leq h_1 \leq h_2 \leq k$,
        previously computed table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}, *)$
          and $AT_{S,I}(S'_{g_2+1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, *)$
Output: table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, J)$ for all $J$
$R_1 \leftarrow S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}$;
$R_2 \leftarrow S'_{g_2+1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}$;
$AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, *) \leftarrow$ undefined;
$Q \leftarrow$ all disks in $\mathcal{D}(r, s)$ of level $j$ intersecting the boundary of $R_1$ or $R_2$;
for all $U \subseteq Q$ such that $|U| \leq 2C'k^2$ do
    if the disks in $I \cup U$ are disjoint then
        for $i = 1$ to 2 do
            $U_i \leftarrow \{D \in U \mid D$ intersects the boundary of $R_i\}$;
            $X_i \leftarrow AT_{S,I}(R_i, U_i)$;
        od;
        $X \leftarrow X_1 \cup X_2 \cup \{D \in U \mid D$ does not intersect the boundary of
            $S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}\}$;
        $J \leftarrow \{D \in U \mid D$ intersects the boundary of $S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}\}$;
        if $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, J)$ is undefined or
          $w(X) > w(AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, J)$ then
            $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, J) \leftarrow X$;
        fi
    fi
od
```

FIG. 2.7. *Computing the auxiliary table* $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, *)$.

which then gives $T_S(I)$. See Figure 2.6 for an example with $k = 4$ in which first the horizontally adjacent rectangles within each row are combined and then the vertically adjacent row rectangles are combined.

The algorithm for computing the table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, *)$ from the table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}, *)$ and $AT_{S,I}(S'_{g_2+1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, *)$ for some $g_1 \leq g_2 < g_3, h_1 \leq h_2$ is shown in Figure 2.7. Let $R_1 = S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}$ and $R_2 = S'_{g_2+1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}$. The algorithm enumerates all independent sets U of disks of level j that intersect the boundary of R_1 or R_2. As each such set has cardinality at most $2C'k^2$, where C' is the constant from Lemma 2.7, there are at most $n^{O(k^2)}$ such sets. If $I \cup U$ is an independent set, the optimal way of extending the set to a larger weight independent set by adding disks of level at least j that are contained in R_1 or R_2 is computed by looking up $AT_{S,I}(R_1, U_1)$ and $AT_{S,I}(R_2, U_2)$, where U_i for $i = 1, 2$ is the set of disks in U that intersect the boundary of R_i. Then the table entry $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, J)$ is updated for the appropriate choice of J provided that the new set is better than the previously stored entry. When all sets U have been processed in this way, the table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, *)$ have their final values.

So far we have assumed that the current j-square S has either no child squares or exactly $(k + 1)^2$ child squares on level $j + 1$. It is easy to handle the case that S has fewer than $(k+1)^2$ child squares on level $j+1$ and possibly some child squares on levels larger than $j + 1$: Just before the computation of T_S, we compute the tables $T_{S'_{g,h}}$ for all $(k+1)^2$ $(j+1)$-squares $S'_{g,h}$ contained in S. For the $(j+1)$-squares that are relevant, the tables are already computed. For every $(j+1)$-square $S'_{g,h}$ that is not relevant, we can enumerate all $n^{O(k^2)}$ sets I of disjoint disks of level at most j that intersect $S'_{g,h}$ and compute $T_{S'_{g,h}}(I)$ by taking the union of the sets $T_{S''}(I^{S''})$ for all child squares S'' of S that are contained in $S'_{g,h}$. Here, $I^{S''} = \{D \in I \mid D \text{ intersects } S''\}$. It is clear that the table entries $T_{S'_{g,h}}(I)$ computed in this way are good provided that good entries of the tables $T_{S''}$ have been computed previously.

This completes the description of the algorithm. In summary, the optimal independent set in $\mathcal{D}(r, s)$ is computed by dynamic programming on the relevant squares using table entries $T_S(I)$, while each such entry is computed by dynamic programming on rectangles within the current square using auxiliary table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}, J)$.

2.4. Running-time of the algorithm.

LEMMA 2.8. *The running-time of the algorithm is $n^{O(k^2)}$ and hence polynomial for any fixed $k > 1$.*

Proof. There are k^2 sets $\mathcal{D}(r, s)$ that have to be considered. As there are at most n disks in $\mathcal{D}(r, s)$, there can be at most n relevant squares. The relevant squares and their forest structure (the links between every relevant square and its children) can be computed easily in time polynomial in n.

For each relevant square S, the algorithm first computes the missing tables $T_{S'_{g,h}}$ for $(j + 1)$-squares $S'_{g,h}$ that are contained in S, as discussed at the end of section 2.3. Each of the $O(k^2)$ such $(j + 1)$-squares can be handled in time $n^{O(k^2)}$. Then the algorithm enumerates $n^{O(k^2)}$ sets I for which a table entry $T_S(I)$ has to be computed. Each such entry is determined by using dynamic programming to fill the table $AT_{S,I}$. Table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}, J)$ are computed for $O(k^2)$ different rectangles $S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}$, and for each rectangle the computation takes time $n^{O(k^2)}$, as can be seen from the pseudocode in Figures 2.4 and 2.7.

Thus, the total running-time can be bounded by

$$k^2 \cdot n^{O(1)} \cdot \left(O(k^2) \cdot n^{O(k^2)} + n^{O(k^2)} \cdot O(k^2) \cdot n^{O(k^2)} \right) = n^{O(k^2)}. \qquad \Box$$

2.5. Correctness of the algorithm.

LEMMA 2.9. *The entries of all the tables T_S and $AT_{S,I}$ computed by the algorithm are good.*

Proof. First, it is clear that the table entries computed by the algorithm satisfy properties (a) and (b) of Definitions 2.4 and 2.6. It remains to show that the sets stored in the tables are indeed of maximum weight.

The proof is by induction on the number of squares processed by the algorithm. Initially, the claim of the lemma holds vacuously. Now assume that the algorithm processes a relevant j-square S and that good entries for the tables $T_{S'}$ have been computed for all relevant squares S' that were processed before S. Let I be a set of disjoint disks of level smaller than j that intersect S.

Consider the computation of the table entries $AT_{S,I}(S'_{g,h}, J)$ as shown in Figure 2.4. Fix some set J of disks at level j intersecting the boundary of $S'_{g,h}$ such that $I \cup J$ is an independent set. Let X^* be a maximum weight set of disks at level at least j that are contained in $S'_{g,h}$ such that $I \cup J \cup X^*$ is an independent set. Let $X^*_{=j}$ be the set of disks at level j in X^* and let $U^* = J \cup X^*_{=j}$. Then U^* is one of the sets U enumerated by the algorithm, and we consider the iteration of the for-loop when this set is processed. Since the entries of $T_{S'_{g,h}}$ are good, the lookup in $T_{S'_{g,h}}(I' \cup U)$ returns a set of weight at least $w(X^* \setminus X^*_{=j})$. Then the set

$$\begin{aligned} X &= T_{S'_{g,h}}(I' \cup U) \cup \{D \in U \mid D \text{ is contained in } S'_{g,h}\} \\ &= T_{S'_{g,h}}(I' \cup U) \cup X^*_{=j} \end{aligned}$$

computed by the algorithm has weight at least $w(X^* \setminus X^*_{=j}) + w(X^*_{=j}) = w(X^*)$. Hence, the table entry $AT_{S,I}(S'_{g,h}, J)$ contains a set X of weight at least $w(X^*)$ when the algorithm of Figure 2.4 terminates. Since $AT_{S,I}(S'_{g,h}, J)$ satisfies properties (a) and (b) of Definition 2.6 and X^* is a maximum weight set with this property, we get that $w(AT_{S,I}(S'_{g,h}, J)) = w(X^*)$. Hence, the computed table entry $AT_{S,I}(S'_{g,h}, J)$ is good.

Consider the computation of a table entry $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, J)$ as shown in Figure 2.7. Assume that the previously computed entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}, *)$ and $AT_{S,I}(S'_{g_2+1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, *)$ are good. Let $R_1 = S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}$ and $R_2 = S'_{g_2+1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}$.

Fix some set J of disks at level j intersecting the boundary of $S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2} = R_1 \cup R_2$ such that $I \cup J$ is an independent set. Let X^* be a maximum weight set of disks at level at least j that are contained in $R_1 \cup R_2$ such that $I \cup J \cup X^*$ is an independent set. Let $X^*_{1,2}$ be the set of disks at level j in X^* that intersect the boundary of R_1 and the boundary of R_2. Let X^*_1 and X^*_2 be the set of disks in X^* that are contained in R_1 and in R_2, respectively.

Let $U^* = J \cup X^*_{1,2}$. Then U^* is one of the sets U enumerated by the algorithm. For this set U^*, the table lookups $AT_{S,I}(R_i, U_i)$ for $i = 1, 2$, with U_i calculated as shown in Figure 2.7, yield disjoint sets X_1 and X_2 of weight at least $w(X^*_1)$ and $w(X^*_2)$, respectively. Then the set

$$\begin{aligned} X &= X_1 \cup X_2 \cup \{D \in U \mid D \text{ does not intersect the boundary of } S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}\} \\ &= X_1 \cup X_2 \cup X^*_{1,2} \end{aligned}$$

calculated by the algorithm has weight at least $w(X^*_1) + w(X^*_2) + w(X^*_{1,2}) = w(X^*)$. Hence, the table entry $AT_{S,I}(R_1 \cup R_2, J)$ contains a set X of weight at least $w(X^*)$ when the algorithm of Figure 2.7 terminates, and is thus a good entry.

So we see that the computed auxiliary table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}, *)$ are indeed good for the rectangles $S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}$. Since the algorithm then sets $T_S(I)$ equal to $AT_{S,I}(S, \emptyset)$, this shows that the computed entries of $T_S(I)$ are good as well. $\square$

We observe that all disks in $\mathcal{D}(r, s)$ are contained in some relevant square without parent, and that all relevant squares without parent are disjoint. By Lemma 2.9, the computed table entries $T_S(\emptyset)$ are good for all relevant squares without parent, and this shows that the algorithm indeed computes an optimal independent set in $\mathcal{D}(r, s)$. Together with the discussion in section 2.1, we have that the algorithm achieves approximation ratio $(1 + \frac{1}{k-1})^2 \leq 1 + \frac{3}{k-1}$. In order to achieve approximation ratio

$1 + \varepsilon$, we can set $k = \lceil 3/\varepsilon \rceil + 1$. The running-time is $n^{O(k^2)}$ and hence polynomial for any fixed $k > 1$. So the algorithm indeed constitutes a PTAS. We summarize our result in the following theorem.

THEOREM 2.10. *There is a PTAS for MWIS in disk graphs, provided that a disk representation of the graph is given. The running-time for achieving approximation ratio $1 + \varepsilon$ is $n^{O(1/\varepsilon^2)}$ for a disk graph with n disks.*

3. A PTAS for vertex cover in disk graphs.

In this section we describe a PTAS for MWVC. The basic approach is similar to the PTAS for MWIS of the previous section, but a number of details cause additional technical difficulties and require a different treatment. One problem is that the size of a vertex cover cannot be bounded by area arguments as the size of an independent set can. The main idea to circumvent this problem is to work with the complements of vertex covers, which are independent sets.

The partitioning of the disks into levels and the subdivision of the plane into squares at each level is the same as for MWIS. Again, all values of r and s such that $0 \le r, s < k$ are considered in turn. Note that the definition of the squares on each level depends on r and s. Contrary to the PTAS for MWIS, disks that are not completely contained in some square on their level are not removed; instead, these disks are considered in all squares on their level that they intersect (there are at most four such squares). Now a j-square S is called relevant if $\mathcal{D}$ contains a disk of level j that intersects S.

Consider some j-square S and let $\mathcal{D}^S$ denote the set of all disks in $\mathcal{D}$ that intersect S. Let $\mathcal{D}^S_{<j}$ be the set of disks in $\mathcal{D}^S$ that have level smaller than j. Define $\mathcal{D}^S_{\le j}$, $\mathcal{D}^S_{=j}$, $\mathcal{D}^S_{\ge j}$, and $\mathcal{D}^S_{>j}$ analogously.

We say that a set $Z \subseteq \mathcal{D}$ is a *pseudocover* of S if, for any two disks $D_1, D_2 \in \mathcal{D}^S$ that intersect in S (i.e., $D_1 \cap D_2 \cap S \neq \emptyset$), Z contains D_1 or D_2 (or both). Note that a pseudocover of S need not contain D_1 or D_2 if the only intersection of D_1 with D_2 is outside S. We observe that, for any two disks in $\mathcal{D}$ that intersect, there exists a relevant square in which they intersect.

For any pseudocover Z of S, call $\mathcal{D}^S_{<j} \cap Z$ the *projection* of Z onto $\mathcal{D}^S_{<j}$. While processing S, the algorithm considers only pseudocovers Z of S for which the projection of Z onto $\mathcal{D}^S_{<j}$ equals $\mathcal{D}^S_{<j} \setminus I$ for some independent set $I \subseteq \mathcal{D}^S_{<j}$. As the number of independent sets in $\mathcal{D}^S_{<j}$ is bounded by $n^{O(k^2)}$ by Lemma 2.5, we can enumerate all of them and thus, by taking the complements, also the corresponding projections of pseudocovers of S onto $\mathcal{D}^S_{<j}$ in time $n^{O(k^2)}$.

As in the PTAS for MWIS, all relevant squares are processed in a bottom-up fashion, and for each square S a table T_S is computed. For a relevant j-square S and a set I of disjoint disks in $\mathcal{D}^S_{<j}$, the table entry $T_S(I)$ has the following property.

PROPERTY 1. *For every relevant j-square S and every set I of disjoint disks in $\mathcal{D}^S_{<j}$, the table entry $T_S(I)$ is a subset of $\mathcal{D}^S_{\ge j}$ such that the set*

$$(\mathcal{D}^S_{<j} \setminus I) \cup T_S(I)$$

is a pseudocover of S.

Unlike in the PTAS for MWIS, the entries $T_S(I)$ will not be optimal (minimum weight) sets with the stated property, but will still be good enough to achieve the desired approximation ratio. It is clear that we cannot expect to compute minimum weight entries $T_S(I)$, as in that case the entry $T_S(\emptyset)$ of a relevant square S without parent would represent an optimal vertex cover of all disks contained in S, thus solving an $\mathcal{NP}$-hard problem optimally. (Note that we do not delete any disks from the given instance of MWVC in disk graphs before we compute the table entries, contrary to our algorithm for MWIS.)

In the end, the algorithm outputs the union of the sets $T_S(\emptyset)$ for all relevant squares S without parent. By the definition of pseudocovers, this union is a vertex cover of $\mathcal{D}$.

At a relevant j-square S, all independent sets I in $\mathcal{D}^S_{<j}$ are enumerated in time $n^{O(k^2)}$. The computation of the table entry $T_S(I)$ for one such set I is described in the following.

Assume again that the j-square S has $(k+1)^2$ child squares on level $j+1$ (with the same justification as in the case of MWIS) and denote these child squares by $S'_{g,h}$, where g is the row index and h is the column index, $0 \le g, h \le k$. Since the algorithm processes the relevant squares in order of nonincreasing levels, the table $T_{S'_{g,h}}$ has already been computed for each such child square $S'_{g,h}$.

In order to determine $T_S(I)$, the algorithm computes an auxiliary table $AT_{S,I}$ with entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}, J)$ for certain values of g_1, g_2, h_1, h_2, where J is a set of disks that are on level j and intersect the boundary of $S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}$ such that $I \cup J$ is an independent set. For a rectangle R, let $\mathcal{D}^{\partial R}$ be the set of all disks in $\mathcal{D}$ that intersect the boundary of R.

PROPERTY 2. *Let S be a relevant j-square and I an independent set in $\mathcal{D}^S_{<j}$. Let $R = S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}$ be a rectangle of child squares of S and J a set of disks of level j intersecting the boundary of R such that $I \cup J$ is an independent set. The table entry $AT_{S,I}(R, J)$ is a set of disks in $\mathcal{D}$ that*

- *either have level j and are contained in R or*
- *have level at least $j+1$ and intersect R*

such that

$$(\mathcal{D}^S_{<j} \setminus I) \cup (\mathcal{D}^{\partial R}_{=j} \setminus J) \cup AT_{S,I}(R, J)$$

is a pseudocover of R.

Once the table entries $AT_{S,I}(S'_{0 \cdot\cdot k, 0 \cdot\cdot k}, J) = AT_{S,I}(S, J)$ for all J are computed, $T_S(I)$ is obtained by taking the set $AT_{S,I}(S, J) \cup (\mathcal{D}^{\partial S}_{=j} \setminus J)$ of minimum weight (over all J).

The table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}, J)$ are again computed by dynamic programming. By Lemma 2.7, for each rectangle $S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}$ there are at most $n^{O(k^2)}$ different sets that are relevant as table index J for $AT_{S,I}(S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}, J)$.

As the basis of the dynamic programming, the table entries $AT_{S,I}(S'_{g \cdot\cdot g, h \cdot\cdot h}, J)$ are computed for $0 \le g, h \le k$ as shown in Figure 3.1. For each child square $S'_{g,h}$, all independent sets U of disks of level j intersecting $S'_{g,h}$ are enumerated in time $n^{O(k^2)}$ (by Lemma 2.5). If $I \cup U$ is an independent set, the table entry $T_{S'_{g,h}}(I' \cup U)$ is looked up, where $I' = \{D \in I \mid D \text{ intersects } S'_{g,h}\}$, in order to obtain the pseudocover of $S'_{g,h}$ given by the set

$$(\mathcal{D}^S_{<j} \setminus I) \cup (\mathcal{D}^{S'_{g,h}}_{=j} \setminus U) \cup T_{S'_{g,h}}(I' \cup U).$$

Then J is taken to be the set of disks in U intersecting the boundary of $S'_{g,h}$, and the table entry $AT_{S,I}(S'_{g,h}, J)$ is updated appropriately if the current pseudocover has smaller weight than the one previously stored. (Of course, if S does not have any child squares, the table lookups $T_{S'_{g,h}}(I' \cup U)$ are taken to return the empty set.)

```
**Input:** square $S$ on level $j$,
           set $I$ of disjoint disks of level $< j$ intersecting $S$,
           integers $g, h$ with $0 \leq g, h \leq k$
**Output:** table entries $AT_{S,I}(S'_{g,h}, J)$ for all $J$
$AT_{S,I}(S'_{g,h}, *) \leftarrow$ undefined;
$Q \leftarrow$ all disks in $\mathcal{D}$ of level $j$ intersecting $S'_{g,h}$;
**for** all $U \subseteq Q$ such that $|U| \leq Ck^2$ **do**
    **if** the disks in $I \cup U$ are disjoint **then**
        $I' \leftarrow \{D \in I \mid D$ intersects $S'_{g,h}\}$;
        $X \leftarrow T_{S'_{g,h}}(I' \cup U)$;
        $X \leftarrow X \cup \{D \in \mathcal{D}^{S'_{g,h}}_{=j} \mid D$ is contained in $S'_{g,h}$ and $D \notin U\}$;
        $J \leftarrow \{D \in U \mid D$ intersects the boundary of $S'_{g,h}\}$;
        **if** $AT_{S,I}(S'_{g,h}, J)$ is undefined **or**
          $w(X) < w(AT_{S,I}(S'_{g,h}, J)$ **then**
            $AT_{S,I}(S'_{g,h}, J) \leftarrow X$;
        **fi**
    **fi**
**od**
```

FIG. 3.1. *Computing the auxiliary table $AT_{S,I}(S'_{g,h}, *)$ for MWVC.*

Next, the information from the tables of two rectangles is combined to obtain the table for the rectangle representing the union of the two rectangles. Again, we discuss only the case where the two rectangles share a horizontal edge and have the same width. The algorithm for computing the table $AT_{S,I}(S'_{g_1\cdot\cdot g_3, h_1\cdot\cdot h_2}, *)$ from the tables $AT_{S,I}(S'_{g_1\cdot\cdot g_2, h_1\cdot\cdot h_2}, *)$ and $AT_{S,I}(S'_{g_2+1\cdot\cdot g_3, h_1\cdot\cdot h_2}, *)$ for some $g_1 \leq g_2 < g_3, h_1 \leq h_2$ is shown in Figure 3.2. Let $R_1 = S'_{g_1\cdot\cdot g_2, h_1\cdot\cdot h_2}$ and $R_2 = S'_{g_2+1\cdot\cdot g_3, h_1\cdot\cdot h_2}$. The algorithm enumerates all independent sets U of disks of level j that intersect the boundary of R_1 or R_2. As each such set has cardinality at most $2C'k^2$, where C' is the constant from Lemma 2.7, there are at most $n^{O(k^2)}$ such sets. If $I \cup U$ is an independent set, the table entries $X_1 = AT_{S,I}(R_1, U_1)$ and $X_2 = AT_{S,I}(R_2, U_2)$, where U_1 and U_2 are calculated as shown in Figure 3.2, are looked up to obtain the pseudocover of $R_1 \cup R_2 = S'_{g_1\cdot\cdot g_3, h_1\cdot\cdot h_2}$ given by the set

$$(\mathcal{D}^S_{<j} \setminus I) \cup ((\mathcal{D}^{\partial R_1}_{=j} \cup \mathcal{D}^{\partial R_2}_{=j}) \setminus U) \cup X_1 \cup X_2.$$

Then the table entry $AT_{S,I}(S'_{g_1\cdot\cdot g_3, h_1\cdot\cdot h_2}, J)$ for

$$J = \{D \in U \mid D \text{ intersects the boundary of } S'_{g_1\cdot\cdot g_3, h_1\cdot\cdot h_2}\}$$

is updated appropriately if this pseudocover has smaller weight than the one previously stored.

In the end, the algorithm outputs the union of the sets $T_S(\emptyset)$, taken over all relevant squares S that do not have a parent. We will see that this is a $(1 + \frac{6}{k})$-approximation of the minimum weight vertex cover.

3.1. The algorithm outputs a vertex cover.

LEMMA 3.1. *The algorithm outputs a vertex cover of $\mathcal{D}$.*

Proof. We prove by induction on the number of processed squares that Properties 1 and 2 hold for all computed table entries. Let S be some j-square. Recall that

```
Input: square $S$ on level $j$,
        set $I$ of disjoint disks of level $< j$ intersecting $S$,
        integers $g_1, g_2, g_3$ with $0 \le g_1 \le g_2 < g_3 \le k$,
        integers $h_1, h_2$ with $0 \le h_1 \le h_2 \le k$,
        previously computed table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}, *)$ and
            $AT_{S,I}(S'_{g_2+1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, *)$
Output: table entries $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, J)$ for all $J$
$R_1 \leftarrow S'_{g_1 \cdot\cdot g_2, h_1 \cdot\cdot h_2}$;
$R_2 \leftarrow S'_{g_2+1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}$;
$AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, *) \leftarrow$ undefined;
$Q \leftarrow$ all disks in $\mathcal{D}$ of level $j$ intersecting the boundary of $R_1$ or $R_2$, i.e.,
        $\mathcal{D}^{\partial R_1}_{=j} \cup \mathcal{D}^{\partial R_2}_{=j}$;
for all $U \subseteq Q$ such that $|U| \le 2C'k^2$ do
    if the disks in $I \cup U$ are disjoint then
        for $i = 1$ to 2 do
            $U_i \leftarrow \{D \in U \mid D$ intersects the boundary of $R_i\}$;
            $X_i \leftarrow AT_{S,I}(R_i, U_i)$;
        od;
        $X \leftarrow X_1 \cup X_2 \cup \{D \in Q \setminus U \mid D$ does not intersect the boundary of
                $S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}\}$;
        $J \leftarrow \{D \in U \mid D$ intersects the boundary of $S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}\}$;
        if $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, J)$ is undefined or
           $w(X) < w(AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, J)$ then
            $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, J) \leftarrow X$;
        fi
    fi
od
```

FIG. 3.2. *Computing the auxiliary table $AT_{S,I}(S'_{g_1 \cdot\cdot g_3, h_1 \cdot\cdot h_2}, *)$ for MWVC.*

$\mathcal{D}^S$ is the set of all disks in $\mathcal{D}$ that intersect S and that $\mathcal{D}^S_{<j}$ denotes the set of disks in $\mathcal{D}^S$ whose level is smaller than j, and similarly for $\mathcal{D}^S_{\le j}$, $\mathcal{D}^S_{=j}$, $\mathcal{D}^S_{\ge j}$, and $\mathcal{D}^S_{>j}$. Let I be a set of disjoint disks in $\mathcal{D}^S_{<j}$.

Assume that the entries of all tables $T_{S'}$ computed for previously processed squares S' satisfy Property 1. Then each set $T_{S'_{g,h}}(I' \cup U)$ that is looked up by the algorithm of Figure 3.1 is such that

$$(\mathcal{D}^{S'_{g,h}}_{\le j} \setminus (I' \cup U)) \cup T_{S'_{g,h}}(I' \cup U)$$

is a pseudocover of $S'_{g,h}$. Therefore, the set stored in $AT_{S,I}(S'_{g,h}, J)$ for $J = \{D \in U \mid D$ intersects the boundary of $S'_{g,h}\}$ satisfies Property 2.

Now assume that the information from table entries $AT_{S,I}(R_1, *)$ and $AT_{S,I}(R_2, *)$ is combined to obtain the table entries $AT_{S,I}(R_1 \cup R_2, *)$ using the algorithm of Figure 3.2. Consider some iteration of the algorithm of Figure 3.2 and let U_1 and U_2 be defined as calculated by the algorithm. Note that $U_1 \cup U_2 = U$ but $U_1 \cap U_2$ can be nonempty. Since table entries $AT_{S,I}(R_i, U_i)$ for $i = 1, 2$ satisfy Property 2, we have that the set

$$(\mathcal{D}^S_{<j} \setminus I) \cup ((\mathcal{D}^{\partial R_1}_{=j} \cup \mathcal{D}^{\partial R_2}_{=j}) \setminus (U_1 \cup U_2)) \cup AT_{S,I}(R_1, U_1) \cup AT_{S,I}(R_2, U_2)$$

is indeed a pseudocover of R_1 and R_2 and thus also of $R_1 \cup R_2$. Therefore, the set stored in $AT_{S,I}(R_1 \cup R_2, J)$ for $J = \{D \in U \mid D \text{ intersects the boundary of } R_1 \cup R_2\}$ satisfies Property 2 as well.

When the auxiliary table entries $AT_{S,I}(S,*)$ have been computed, the algorithm obtains $T_S(I)$ by taking the minimum weight set obtainable as $AT_{S,I}(S,J) \cup (\mathcal{D}^{\partial S}_{=j} \setminus J)$. Since the table entry $AT_{S,I}(S,J)$ leading to the minimum satisfies Property 2, the resulting table entry $T_S(I)$ satisfies Property 1.

In the end, the algorithm outputs a union of pseudocovers $T_S(\emptyset)$ in all relevant squares S without parent. As every intersection of two disks is contained in some relevant square without parent, the algorithm always outputs a vertex cover of $\mathcal{D}$. ☐

3.2. Analysis of approximation ratio. Let $\mathcal{C}$ be any optimal vertex cover of $\mathcal{D}$. Every disk hits at most one vertical line on its level and at most one horizontal line on its level. For any pair of values r and s, $0 \le r, s < k$, let $\mathcal{C}(r,s)$ be the set of all disks in $\mathcal{C}$ that hit a vertical line on their level whose index modulo k equals r or a horizontal line on their level whose index modulo k equals s. Note that there must be a value of r such that the total weight of disks in $\mathcal{C}$ that hit a vertical line on their level whose index modulo k equals r is at most $\frac{1}{k}w(\mathcal{C})$. An analogous argument shows that there is a value of s with the corresponding property for horizontal lines. Therefore, there exist values of r and s such that the total weight of $\mathcal{C}(r,s)$ is at most $\frac{2}{k}$ times the weight of $\mathcal{C}$. Consider the subdivision of the plane that results from this choice of r and s.

Let $\mathcal{R}$ be the set of all relevant squares. For any j-square S, let $\mathcal{C}(S)$ denote the disks in $\mathcal{C}$ that intersect S and that are on level j. Note that $\mathcal{C} = \bigcup_{S \in \mathcal{R}} \mathcal{C}(S)$ but the sets $\mathcal{C}(S)$ and $\mathcal{C}(S')$ for $S \neq S'$ need not be disjoint.

LEMMA 3.2. *We have* $\sum_{S \in \mathcal{R}} w(\mathcal{C}(S)) \le (1 + \frac{6}{k}) w(\mathcal{C})$.

Proof. The only disks in $\mathcal{C}$ that are counted more than once in the sum on the left-hand side are those that intersect several squares on their level. However, any disk can intersect at most 4 squares on its level. Furthermore, only disks in $\mathcal{C}(r,s)$ can intersect more than 1 square on their level. Thus, only disks of total weight $w(\mathcal{C}(r,s)) \le \frac{2}{k}w(\mathcal{C})$ are counted multiple times, and each disk is counted at most four times on the left-hand side, while it is counted once in $w(\mathcal{C})$. Thus $\sum_{S \in \mathcal{R}} w(\mathcal{C}(S))$ can exceed $w(\mathcal{C})$ by at most $3 \cdot \frac{2}{k} w(\mathcal{C})$, establishing the claimed inequality. ☐

Lemma 3.2 shows that the sum of the weights of the sets $\mathcal{C}(S)$ is only slightly larger than the weight of $\mathcal{C}$, although certain disks are counted several times in this sum. The following lemma shows that the weight of the vertex cover output by our algorithm does not exceed this sum. Intuitively, the lemma means that, on average, for each square S on some level j the weight of the set of disks from $\mathcal{D}^S_{=j}$ chosen by the algorithm does not exceed the weight of the disks chosen from $\mathcal{D}^S_{=j}$ in the optimal solution.

LEMMA 3.3. *Let A be the vertex cover output by the algorithm. Then* $w(A) \le \sum_{S \in \mathcal{R}} w(\mathcal{C}(S))$.

Proof. For any relevant j-square S or any rectangle R of child squares of S, let $\mathcal{C}^S$ and $\mathcal{C}^R$ be the set of all disks in $\mathcal{C}$ that intersect S and R, respectively. Define $\mathcal{C}^S_{<j}$, $\mathcal{C}^S_{\le j}$, $\mathcal{C}^S_{=j}$, etc. as usual. Observe that $\mathcal{C}(S) = \mathcal{C}^S_{=j}$. We claim that, after T_S has been computed, we have

$$w(T_S(\mathcal{D}^S_{<j} \setminus \mathcal{C}^S_{<j})) \le \sum_{S' : S' \prec S} w(\mathcal{C}(S')), \tag{3.1}$$

where $S' \prec S$ means that S' is a relevant square that is contained in S. Note that $S \prec S$.

The proof is by induction on the number of relevant squares that have been processed by the algorithm. Assume that the algorithm is about to process the relevant square S and that (3.1) holds for all squares that have been processed before S. One of the independent sets I in $\mathcal{D}^S_{<j}$ enumerated by the algorithm is equal to $\mathcal{D}^S_{<j} \setminus \mathcal{C}^S_{<j}$. Consider that set I in the following.

We turn to the computation of the auxiliary table entries $AT_{S,I}(R, J)$, where $R = S'_{g_1 \cdots g_2, h_1 \cdots h_2}$ is a rectangle of subsquares of S. We claim that when the table entries $AT_{S,I}(R, *)$ are computed, we have

$$w(AT_{S,I}(R,J)) \ \leq \ w(\{D \in \mathcal{C}^R_{=j} \mid D \text{ is contained in } R\}) + \textstyle\sum_{S':S' \prec R, S' \neq S} w(\mathcal{C}(S'))$$
$$\text{for } J = \{D \in \mathcal{D}^R_{=j} \setminus \mathcal{C} \mid D \text{ intersects the boundary of } R\} = \mathcal{D}^{\partial R}_{=j} \setminus \mathcal{C}. \tag{3.2}$$

Note that the sum in (3.2) is over all relevant j'-squares, $j' > j$, that are contained in R, and that the sum does *not* include the term $w(\mathcal{C}(S))$ even if $R = S$.

First, consider the computation of $AT_{S,I}(S'_{g,h}, *)$ by the algorithm of Figure 3.1. In one of the iterations of the for-loop, the set U is equal to $Q \setminus \mathcal{C} = \mathcal{D}^{S'_{g,h}}_{=j} \setminus \mathcal{C}$. For this set U, the table entry $T_{S'_{g,h}}(I' \cup U) = T_{S'_{g,h}}(\mathcal{D}^{S'_{g,h}}_{\leq j} \setminus \mathcal{C}^{S'_{g,h}}_{\leq j})$ is looked up. Since (3.1) holds for $T_{S'_{g,h}}$, we get that $w(T_{S'_{g,h}}(I' \cup U)) \leq \sum_{S':S' \prec S'_{g,h}} w(\mathcal{C}(S'))$. Hence, the set

$$X = T_{S'_{g,h}}(I' \cup U) \cup \{D \in \mathcal{D}^{S'_{g,h}}_{=j} \setminus U \mid D \text{ is contained in } S'_{g,h}\}$$

stored in $AT_{S,I}(S'_{g,h}, J)$ for

$$\begin{aligned} J &= \{D \in U \mid D \text{ intersects the boundary of } S'_{g,h}\} \\ &= \{D \in \mathcal{D}^{S'_{g,h}}_{=j} \setminus \mathcal{C} \mid D \text{ intersects the boundary of } S'_{g,h}\} \end{aligned}$$

has weight at most

$$\begin{aligned} &\sum_{S':S' \prec S'_{g,h}} w(\mathcal{C}(S')) + w(\{D \in \mathcal{D}^{S'_{g,h}}_{=j} \setminus U \mid D \text{ is contained in } S'_{g,h}\}) \\ = &\sum_{S':S' \prec S'_{g,h}} w(\mathcal{C}(S')) + w(\{D \in \mathcal{C}^{S'_{g,h}}_{=j} \mid D \text{ is contained in } S'_{g,h}\}). \end{aligned}$$

From this we see that (3.2) is satisfied for $R = S'_{g,h}$.

Next, consider the combination of table entries $AT_{S,I}(R_1, *)$ and $AT_{S,I}(R_2, *)$ to obtain table entries $AT_{S,I}(R_1 \cup R_2, *)$ using the algorithm of Figure 3.2. In one of the iterations of the for-loop, the set U is equal to the set

$$Q \setminus \mathcal{C} = \{D \in \mathcal{D}^{R_1 \cup R_2}_{=j} \setminus \mathcal{C} \mid D \text{ intersects the boundary of } R_1 \text{ or } R_2\} = (\mathcal{D}^{\partial R_1}_{=j} \cup \mathcal{D}^{\partial R_2}_{=j}) \setminus \mathcal{C}.$$

For this set U, the table entries $AT_{S,I}(R_i, U_i)$ are looked up for $i = 1, 2$, where $U_i = \{D \in U \mid D \text{ intersects the boundary of } R_i\}$. Thus, we have $U_i = \mathcal{D}^{\partial R_i}_{=j} \setminus \mathcal{C}$ and we know that (3.2) holds for table entries $AT_{S,I}(R_i, U_i)$. Then the weight of the set

$$\begin{aligned}X &= AT_{S,I}(R_1,U_1) \cup AT_{S,I}(R_2,U_2) \cup \{D \in (\mathcal{D}_{=j}^{\partial R_1} \cup \mathcal{D}_{=j}^{\partial R_2}) \setminus U \mid D \text{ is contained}\\ &\quad \text{in } R_1 \cup R_2\}\\ &= AT_{S,I}(R_1,U_1) \cup AT_{S,I}(R_2,U_2) \cup \{D \in \mathcal{C} \cap (\mathcal{D}_{=j}^{\partial R_1} \cup \mathcal{D}_{=j}^{\partial R_2}) \mid D \text{ is contained}\\ &\quad \text{in } R_1 \cup R_2\}\\ &= AT_{S,I}(R_1,U_1) \cup AT_{S,I}(R_2,U_2) \cup \{D \in \mathcal{C}_{=j}^{\partial R_1} \cup \mathcal{C}_{=j}^{\partial R_2} \mid D \text{ is contained in } R_1 \cup R_2\}\\ &= AT_{S,I}(R_1,U_1) \cup AT_{S,I}(R_2,U_2) \cup \{D \in \mathcal{C}_{=j}^{\partial R_1} \cap \mathcal{C}_{=j}^{\partial R_2} \mid D \text{ is contained in } R_1 \cup R_2\}\\ &= AT_{S,I}(R_1,U_1) \cup AT_{S,I}(R_2,U_2) \cup \{D \in \mathcal{C}_{=j}^{\partial R_1 \cap \partial R_2} \mid D \text{ is contained in } R_1 \cup R_2\}\end{aligned}$$

is at most

$$\begin{aligned}&w(\{D \in \mathcal{C}_{=j}^{R_1} \mid D \text{ is contained in } R_1\}) + \sum_{S' : S' \prec R_1} w(\mathcal{C}(S'))\\ &\quad + w(\{D \in \mathcal{C}_{=j}^{R_2} \mid D \text{ is contained in } R_2\}) + \sum_{S' : S' \prec R_2} w(\mathcal{C}(S'))\\ &\quad + w(\{D \in \mathcal{C}_{=j}^{\partial R_1 \cap \partial R_2} \mid D \text{ is contained in } R_1 \cup R_2\})\\ &= \left(\sum_{S' : S' \prec (R_1 \cup R_2), S' \neq S} w(\mathcal{C}(S')) \right) + w(\{D \in \mathcal{C}_{=j}^{R_1 \cup R_2} \mid D \text{ is contained in } R_1 \cup R_2\}).\end{aligned}$$

The set X is stored in $AT_{S,I}(R_1 \cup R_2, J)$ for $J = \{D \in U \mid D$ intersects the boundary of $R_1 \cup R_2\} = \{D \in \mathcal{D}_{=j}^{R_1 \cup R_2} \setminus \mathcal{C} \mid D$ intersects the boundary of $R_1 \cup R_2\}$ provided its weight is smaller than that of the previously stored set. This shows that (3.2) holds for $R = R_1 \cup R_2$ when the algorithm of Figure 3.2 terminates. Thus, by induction we see that all table entries $AT_{S,I}(R, J)$ satisfy (3.2).

This implies that for $J = \mathcal{D}_{=j}^{\partial S} \setminus \mathcal{C}$, the table entry $AT_{S,I}(S, J)$ has weight at most $w(\{D \in \mathcal{C}_{=j}^{S} \mid D \text{ is contained in } S\}) + \sum_{S' : S' \prec S, S' \neq S} w(\mathcal{C}(S'))$. Note that $\mathcal{D}_{=j}^{\partial S} \setminus J = \mathcal{C}_{=j}^{\partial S}$. The set $AT_{S,I}(S, J) \cup (\mathcal{D}_{=j}^{\partial S} \setminus J)$, which is one of the candidates for the table entry $T_S(I)$, has weight at most $w(\mathcal{C}(S)) + \sum_{S' : S' \prec S, S' \neq S} w(\mathcal{C}(S'))$. Thus, we get that (3.1) holds for T_S as well, and the inductive step with respect to the tables T_S is established.

Finally, let $\mathcal{R}_0$ be the set of all relevant squares without parent. The algorithm outputs $\bigcup_{S \in \mathcal{R}_0} T_S(\emptyset)$ as a solution. For any relevant j-square $S \in \mathcal{R}_0$, we have $\mathcal{C}_{<j}^{S} = \emptyset$. Thus we get from (3.1) that $\sum_{S \in \mathcal{R}_0} w(T_S(\emptyset)) \leq \sum_{S \in \mathcal{R}} w(\mathcal{C}(S))$. □

Combining these lemmas, we get that the algorithm outputs a vertex cover whose weight is at most a factor of $1 + \frac{6}{k}$ larger than the weight of the optimal vertex cover. Furthermore, by a similar analysis as in the case of MWIS, the running-time of the algorithm is bounded by $n^{O(k^2)}$. Thus we obtain our second main theorem.

THEOREM 3.4. *There is a PTAS for MWVC in disk graphs, provided that a disk representation of the graph is given. The running-time for achieving approximation ratio* $1 + \varepsilon$ *is* $n^{O(1/\varepsilon^2)}$ *for a disk graph with* n *disks.*

4. Extension to other geometric intersection graphs.

In the previous sections we have presented PTASs for MWIS and MWVC in the intersection graphs of disks with arbitrary diameters. Our approach does not make use of any specific properties of disks; it is required only that the given geometric objects can be partitioned into levels such that only a constant number (for fixed k) of disjoint objects of level smaller than j can intersect a square on level j. Therefore, the same approach can be used for other geometric objects such as squares or regular polygons. We can also deal with rectangles if the ratio between the height and the width does not differ by more than a constant factor between different rectangles, because it suffices to scale the input along one axis so that the resulting rectangles are approximately squares.

Furthermore, the approach works for geometric objects in any d-dimensional space provided that d is a constant. The space is partitioned into d-dimensional cubes on each level, and instead of removing objects (in the case of MWIS) that hit certain horizontal or vertical lines, we remove objects that hit certain hyperplanes.

More specifically, to obtain PTASs for MWIS and MWVC in the intersection graphs of geometric objects in d dimensions, the following conditions are sufficient:

(i) d is a constant.

(ii) For each object i, a d-dimensional ball B_i that contains i can be determined in polynomial time.

(iii) Let level j contain all given objects whose balls B_i have diameter d_i satisfying $(k+1)^{-j} \geq d_i > (k+1)^{-(j+1)}$. Then the number of disjoint objects of level smaller than j that can intersect a d-dimensional cube with side length $k(k+1)^{-j}$ must be bounded by a constant (for fixed k), and the number of disjoint objects of level j that can intersect the boundary of a d-dimensional hyperrectangle with sides of length at most $k(k+1)^{-j}$ must be bounded by a constant (for fixed k).

(iv) One can decide in polynomial time whether two of the given objects intersect and whether an object intersects an arbitrary cube.

For example, our approach yields a PTAS for MWIS or MWVC in the intersection graphs of n balls in d-dimensional space with running-time $n^{O(1/\varepsilon^{2d-2})}$ for any constant d.

5. Conclusion and open problems.

We have presented the first known PTASs for MWIS and MWVC in the intersection graphs of disks. Our algorithms partition the given disks into levels and apply a shifting strategy on all levels simultaneously. The PTASs can be generalized to other "disk-like" geometric objects in d dimensions, achieving running-time $n^{O(1/\varepsilon^{2d-2})}$ for any constant d. Another PTAS for MWIS in disk graphs, based on ideas similar to ours but using shifted quadtrees in the recursive subdivision strategy, was discovered recently by Chan [8]. Chan's algorithm achieves running-time $n^{O(1/\varepsilon)}$ for MWIS in disk graphs and $n^{O(1/\varepsilon^{d-1})}$ for the d-dimensional generalization. MWVC is not considered in Chan's paper.

Our approximation schemes use a partitioning of the plane into squares on each level, and this works only if the ratio of the height to the width is roughly the same for all given geometric objects. It is not clear whether the approach can be extended to the intersection graphs of arbitrary axis-parallel rectangles, for example. For computing a maximum independent set among n given rectangles, an $O(\log n)$-approximation algorithm was presented by Agarwal, van Kreveld, and Suri [1]. Berman et al. [4] improved upon this result and gave a family of approximation algorithms with approximation ratio $1 + \frac{\log n}{c}$ for *any* positive constant c. It is an open problem to devise an approximation algorithm with constant approximation ratio or even a PTAS for this problem, or to provide evidence that MWIS in intersection graphs of arbitrary rectangles is substantially harder to approximate than in intersection graphs of squares or disks.

Concerning disk graphs, it would be interesting to see whether one can also find a PTAS for the minimum dominating set problem, where the goal is to select a minimum number of disks such that each given disk is either selected or intersects a disk that is selected. For planar graphs and for unit disk graphs, PTASs for the minimum dominating set problem have been found using the shifting strategy [2, 18], but we have not yet been able to adapt the approach of the present paper to the minimum dominating set problem.

The PTASs for disk graphs require that the disk representation of the graph is given as part of the input. It would be interesting to determine whether a PTAS can be obtained even in the case without given representation. In this context we note that for the maximum clique problem in unit disk graphs, which can be solved in polynomial time [9], an exact algorithm that does not need the representation has recently been found by Raghavan and Spinrad [26]. This is somewhat surprising since it is $\mathcal{NP}$-hard to determine whether a given graph is a unit disk graph [7]. In fact, the algorithm by Raghavan and Spinrad is *robust* in the sense that for any given graph G, the algorithm either finds a maximum clique in G or asserts correctly that G is not a unit disk graph. Robust algorithms solving MWIS optimally in certain classes of graphs that are characterized by forbidden subgraphs have been presented by Brandstädt [5]. The concept of robustness can be applied to approximation algorithms for graph problems by calling an algorithm a *robust ρ-approximation algorithm* for a graph class $\mathcal{C}$ if, for any input graph G, the algorithm either outputs a solution that is within a factor ρ of the optimal solution or asserts correctly that G is not a member of $\mathcal{C}$. Recently, a robust PTAS that does not require the disk representation as part of the input has been obtained for MWIS in unit disk graphs by Nieberg, Hurink, and Kern [25].

REFERENCES

[1] P. K. Agarwal, M. van Kreveld, and S. Suri, *Label placement by maximum independent set in rectangles*, Comput. Geom. 11 (1998), pp. 209–218.

[2] B. S. Baker, *Approximation algorithms for NP-complete problems on planar graphs*, J. ACM, 41 (1994), pp. 153–180.

[3] R. Bar-Yehuda and S. Even, *A local-ratio theorem for approximating the weighted vertex cover problem*, Ann. Discrete Math., 25 (1985), pp. 27–45.

[4] P. Berman, B. DasGupta, S. Muthukrishnan, and S. Ramaswami, *Improved approximation algorithms for rectangle tiling and packing*, in Proceedings of the 12th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'01), ACM, New York, SIAM, Philadelphia, 2001, pp. 427–436.

[5] A. Brandstädt, *On robust algorithms for the maximum weight stable set problem*, in Proceedings of the 13th International Symposium on Fundamentals of Computation Theory (FCT 2001), Lecture Notes in Comput. Sci. 2138, Springer-Verlag, Berlin, 2001, pp. 445–458.

[6] A. Brandstädt, V. B. Le, and J. P. Spinrad, *Graph Classes: A Survey*, SIAM Monogr. Discrete Math. Appl. 3, SIAM, Philadelphia, 1999.

[7] H. Breu and D. G. Kirkpatrick, *Unit disk graph recognition is NP-hard*, Comput. Geom. 9 (1998), pp. 3–24.

[8] T. M. Chan, *Polynomial-time approximation schemes for packing and piercing fat objects*, J. Algorithms, 46 (2003), pp. 178–189.

[9] B. N. Clark, C. J. Colbourn, and D. S. Johnson, *Unit disk graphs*, Discrete Math., 86 (1990), pp. 165–177.

[10] S. Doddi, M. V. Marathe, A. Mirzaian, B. M. E. Moret, and B. Zhu, *Map labeling and its generalizations*, in Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'97), ACM, New York, SIAM, Philadelphia, 1997, pp. 148–157.

[11] S. Doddi, M. V. Marathe, and B. M. E. Moret, *Point set labeling with specified positions*, Internat. J. Comput. Geom. Appl., 12 (2002), pp. 29–66.

[12] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[13] W. K. Hale, *Frequency assignment: Theory and applications*, Proceedings of the IEEE, 68 (1980), pp. 1497–1514.

[14] J. Håstad, *Clique is hard to approximate within $n^{1-\epsilon}$*, Acta Math., 182 (1999), pp. 105–142.

[15] J. Håstad, *Some optimal inapproximability results*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC'97), ACM Press, New York, 1997, pp. 1–10.

[16] P. Hliněný and J. Kratochvíl, *Representing graphs by disks and balls*, Discrete Math., 229 (2001), pp. 101–124.

[17] D. S. Hochbaum and W. Maass, *Approximation schemes for covering and packing problems in image processing and VLSI*, J. ACM, 32 (1985), pp. 130–136.

[18] 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 (1998), pp. 238–274.

[19] K. Jansen, *Approximate strong separation with application in fractional graph coloring and preemptive scheduling*, Theoret. Comput. Sci., 302 (2003), pp. 239–256.

[20] K. Jansen and L. Porkolab, *On preemptive resource constrained scheduling: Polynomial-time approximation schemes*, in Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization (IPCO), Lecture Notes in Comput. Sci. 2337, Springer-Verlag, Berlin, 2002, pp. 329–349.

[21] P. Koebe, *Kontaktprobleme der konformen Abbildung*, Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften, Leipzig, Math.-Phys. Klasse, 88 (1936), pp. 141–164.

[22] E. Malesińska, *Graph-Theoretical Models for Frequency Assignment Problems*, Ph.D. thesis, Technische Universität Berlin, Berlin, Germany, 1997.

[23] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz, *Simple heuristics for unit disk graphs*, Networks, 25 (1995), pp. 59–68.

[24] T. A. McKee and F. R. McMorris, *Topics in Intersection Graph Theory*, SIAM Monogr. Discrete Math. Appl. 2, SIAM, Philadelphia, 1999.

[25] T. Nieberg, J. L. Hurink, and W. Kern, *A robust PTAS for maximum independent sets in unit disk graphs*, in Proceedings of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'04), Lecture Notes in Comput. Sci. 3353, Springer-Verlag, New York, 2004, pp. 214–221.

[26] V. Raghavan and J. Spinrad, *Robust algorithms for restricted domains*, in Proceedings of the 12th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'01), ACM, New York, SIAM, Philadelphia, 2001, pp. 460–467.

[27] M. van Kreveld, T. Strijk, and A. Wolff, *Point labeling with sliding labels*, Comput. Geom., 13 (1999), pp. 21–47.

[28] A. Wolff and T. Strijk, *The map-labeling bibliography*, http://i11www.ilkd.uni-karlsruhe.de/∼awolff/map-labeling/bibliography/ (2003).