JOURNAL OF ALGORITHMS **26,** 238–274 (1998)
ARTICLE NO. AL970903

NC-Approximation Schemes for NP- and PSPACE-Hard Problems for Geometric Graphs*

Harry B. Hunt III[†, ‡]

Department of Computer Science, State University of New York, Albany, New York 12222

Madhav V. Marathe[‡, ‖]

Los Alamos National Laboratory, P.O. Box 1663, MS B265, Los Alamos, New Mexico 87544

Venkatesh Radhakrishnan[¶]

Hewlett-Packard Company, 19447 Pruneridge Avenue, Cupertino, California 94014

and

S. S. Ravi,[†] Daniel J. Rosenkrantz,[†, §] and Richard E. Stearns[†, ‡]

Department of Computer Science, State University of New York, Albany, New York 12222

Received February 21, 1995; revised September 10, 1997

We present NC-approximation schemes for a number of graph problems when restricted to geometric graphs including unit disk graphs and graphs drawn in a civilized manner. Our approximation schemes exhibit the same time versus performance trade-off as the best known approximation schemes for planar graphs. We also define the concept of λ-precision unit disk graphs and show that for such

* A preliminary version of this paper appeared in ‘‘Proc. 2nd European Symposium on Algorithms (ESA’94) (pp. 424–435, Springer-Verlag, Berlin/New York).

[†] E-mail: {hunt,ravi,djr,res}@cs.albany.edu.

[‡] Supported by NSF Grant CCR-89-03319 and CCR-94-06611.

[§] Supported by NSF Grant CCR-90-06396.

[¶] E-mail: rven@cup.hp.com.

[‖] Part of the work was done when the author was at SUNY, Albany. E-mail: marathe@lanl.gov. Work supported by Department of Energy Contract W-7405-ENG-36.

graphs the approximation schemes have a better time versus performance trade-off than the approximation schemes for arbitrary unit disk graphs. Moreover, compared to unit disk graphs, we show that for λ-precision unit disk graphs many more graph problems have efficient approximation schemes.

Our NC-approximation schemes can also be extended to obtain efficient NC-approximation schemes for several PSPACE-hard problems on unit disk graphs specified using a restricted version of the hierarchical specification language of Bentley, Ottmann, and Widmayer. The approximation schemes for hierarchically specified unit disk graphs presented in this paper are among the first approximation schemes in the literature for natural PSPACE-hard optimization problems.

Key Words: approximation schemes; parallel algorithms; geometric graphs; unit disk graphs; graphs drawn in a civilized manner; VLSI design; hierarchical specifications.

1. INTRODUCTION

An undirected graph is a *unit disk graph* if its vertices can be put in one-to-one correspondence with circles of equal radius in the plane in such a way that two vertices are joined by an edge if and only if the corresponding circles intersect. (Throughout this paper, tangent circles are assumed to intersect. Without loss of generality, it is assumed that the radius of each disk is 1.) Unit disk graphs have been used to model problems in diverse areas such as broadcast networks [Ha80, Ka84, YWS84], image processing [HM85], VLSI circuit design [MC80], and optimal facility location [WK88, MS84]. Consequently, the complexity of optimization problems for unit disk graphs has been studied extensively in the literature [CCJ90, FPT81, MB + 95, MS84, WK88]. As pointed out in [CCJ90], unit disk graphs need not be perfect since any odd cycle of length 5 or greater is a unit disk graph. Similarly, unit disk graphs need not be planar; in particular, any clique of size 5 or more is a unit disk graph. Thus many of the known efficient algorithms for perfect graphs and planar graphs do not apply to unit disk graphs.

It has been shown in [CCJ90, FPT81, MS84, WK88] that several standard graph theoretic problems are strongly NP-hard *even* when restricted to unit disk graphs. Given this apparent intractability, we investigate whether these problems have efficient approximation algorithms and approximation schemes. Recall that an approximation algorithm for an optimization problem Π provides a *performance guarantee* of ρ if, for every instance I of Π, the solution value returned by the approximation algorithm is within a factor ρ of the optimal value for I. A *polynomial time approximation scheme* (PTAS) for problem Π is a polynomial time approximation algorithm which, given any instance I of Π and an $\epsilon > 0$, returns a solution which is within a factor $(1 + \epsilon)$ of the optimal value for I. An

NC-approximation scheme is an approximation scheme which takes polylog time while using only a polynomial number of processors. A polynomial time approximation scheme whose running time is polynomially dependent on the size of the input and $1/\epsilon$ is called a *fully polynomial time approximation scheme*. Since the problems considered in this paper are strongly NP-hard, we cannot hope to devise fully polynomial time approximation schemes for them (see [GJ79], Theorem 6.8)]. However, this negative result does not preclude the existence of polynomial time approximation schemes for these problems. Such polynomial time approximation schemes for strongly NP-hard problems are quite rare in the literature.

Here, we present efficient approximations and approximation schemes for NP- and PSPACE-hard problems when restricted to geometric intersection graphs, particularly unit disk graphs. Our results also apply to graphs drawn in a civilized manner (see Definition 3.1). Our approximation schemes can also be extended so as to apply to several PSPACE-hard optimization problems on unit disk graphs presented hierarchically using a restricted form of the hierarchical input language (HIL) of Bentley, Ottmann, and Widmayer [BOW83]. All of our algorithms assume that a geometric representation of the graph is given as input. Thus, for example, in case of unit disk graphs we assume that the graph is specified by a set of unit disks in the plane. This assumption about input representation is both reasonable and realistic. It is reasonable since the recognition problem for unit disk graphs is NP-hard; i.e., given a graph specified as set of vertices and edges, it is NP-hard to tell whether the graph can be realized as the intersection graph of a set of unit disks. The assumption about input representation is also realistic, since for most applications the graph is naturally specified using the intersection model. For example, the problem of finding a minimum dominating set for unit disk graphs arises in the context of broadcast networks [CCJ90]. In this application, each transmitter is specified as a unit disk in the plane. Similarly, graphs drawn in a civilized manner arise naturally in the context of mesh generation and efficient mapping of problem structure onto parallel machines [Te91].

The approximation algorithms for problems restricted to unit disk graphs also extend when the unit disks are specified hierarchically. Hierarchical specifications derive their motivation from the design and analysis of VLSI circuits. Although such circuits can be made up of millions of components, they often have a highly regular structure. This regular structure often makes it possible for their design to be specified succinctly using hierarchical specifications. Our primary motivation for studying hierarchically specified intersection graphs is that many VLSI design specification languages such as Caltech Intermediate Form (CIF) use hierarchical collections of geometric objects such as circles, rectangles, and other polygonal figures as primitives to represent large designs [MC80]. Hence, it is natural to investigate the complexity of graph theoretic problems for intersection graphs specified hierarchically. In the past, several other authors have proposed hierarchical specifications (see [LW92] for details). To avoid any ambiguity, we refer to hierarchical specifications of Bentley *et al.* [BOW83] as BOW-specifications throughout this paper. BOW-specifications use a subset of CIF to define a set of geometric objects and thus they can be interpreted naturally as specifying the intersection graph of the set of objects defined. It is in this sense that we view BOW-specifications as specifying geometric intersection graphs.

In practice, it is difficult to process designs specified using the general form of BOW-specifications since even simple questions such as ‘‘Is there a pair of intersecting rectangles in the set?’’ are NP-hard for such designs. Hence, Bentley *et al.* [BOW83] also proposed a restricted form of BOW-specifications called *consistent specifications* and showed that several standard problems become tractable for consistent BOW-specifications. For instance, the question of whether there exists a pair of intersecting rectangles in the given set is polynomially solvable for such descriptions. Unfortunately, as pointed out in [BOW83], consistency is a very strong restriction and no real designs can be specified using consistent BOW-specifications. Bentley *et al.* state that: ‘‘It will be important to identify families of restrictions that exclude only a few designs but admit very rapid processing of remaining designs.’’ As a step toward identifying such restrictions, we define a family of restricted BOW-specifications called the *k-near-consistent BOW-specifications*. In [MR + 97], we proved that a number of graph theoretic problems are PSPACE-hard when instances are specified using 1-near consistent BOW-specifications.[1] These hardness results might suggest that k-near-consistent specifications may not be amenable to rapid processing. Although this is true if we wish to solve the problems exactly, we show that several of these PSPACE-hard problems possess efficient approximation algorithms and approximation schemes.

We now summarize the main contributions of this paper.

1. We present approximation schemes (sequential and parallel) for several natural graph problems, when restricted to unit disk graphs or graphs drawn in a civilized manner. Previously, no such approximation schemes were known for problems on unit disk graphs. Our approximation schemes can be extended to geometric intersection graphs, both of other regular polygons and also of regular geometric objects in higher dimensions.

2. Our approximation schemes for λ-precision unit disk graphs (see Definition 3.2) and for graphs drawn in a civilized manner (see Definition

[1] In [MR + 97], 1-near-consistent specifications are called 1-level-restricted specifications.

3.1) have the same time versus performance trade-off as the approximation schemes for planar graph problems given in [Ba94].

3. We also present approximations and approximation schemes for unit disk graphs specified using k-near-consistent BOW-specifications. Many of the problems shown here to have efficient approximation schemes are PSPACE-complete (see [MR + 97]). Thus the approximation schemes presented here along with those in [MH + 94] are the first approximation schemes for *natural* PSPACE-hard optimization problems. The question of whether there exist approximation schemes for natural PSPACE-hard problems was raised by Condon *et al.* [CF + 93].

4. Our definition of near-consistent BOW-specifications (see Definition 6.2) is a step toward solving the general problem posed by Bentley *et al.* [BOW83] of finding sufficient syntactic restrictions on BOW-specifications, which allow for rapid processing of the designs and also include many realistic designs.

2. RELATED WORK

The complexity of finding exact solutions to graph problems, when restricted to unit disk graphs, has been studied extensively in [CCJ90, FPT81, MS84, WK88]. In [MB + 95], we showed that several natural graph problems such as maximum independent set, minimum vertex cover, and minimum dominating set can be approximated to within a constant factor of the optimum for unit disk graphs specified using a *graph theoretic* representation. Other researchers have also studied the existence of efficient approximation algorithms for coloring unit disk graphs (see [GSW94] and [Pe91]).

The concept of λ-precision unit disk graphs (see Definition 3.2) bears a close resemblance to the concept of intersection graph for a k-neighborhood system defined by Miller *et al.* [MT + 97]. The *neighborhood* of a point p in R^d is a closed ball of a certain radius centered at p. A k-neighborhood system in R^d is a collection of n neighborhoods such that no ball contains more than k centers. It can be seen that every λ-precision unit disk graph is the intersection graph of a k-neighborhood system in R^2, where k depends on the precision factor λ. Similarly, the intersection graph of a k-neighborhood system in R^2 with unit balls is a λ-precision unit disk graph, where λ is the minimum distance between the centers of any two balls. Using the geometric separator concept of Eppstein *et al.* [EMT93, MT + 97], one can find an approximation scheme for problems restricted to λ-precision unit disk graphs in the same fashion as the planar separator theorem [LT79] was used to find approximation schemes for planar graph problems. However, this approach has two main drawbacks. The first is that as in the case of planar graphs, the approximation schemes apply only in the asymptotic sense and hence are not practical (see [Ba94]). The second drawback is that problems such as maximum independent set and minimum dominating set for which approximation schemes can be designed for arbitrary unit disk graphs by our method *cannot* be solved at all by the separator approach.[2] This is because an arbitrary unit disk graph on n nodes can have a clique of size n. Hence, in general, unit disk graphs do not have "good" separators.

In [Ba94], polynomial time approximation schemes were provided for a large class of problems on planar graphs. Recently, several researchers [KS93, DST96, HM + 93] showed how to parallelize the ideas in [Ba94] to obtain efficient NC-approximation schemes for problems restricted to planar graphs. A number of other researchers have used ideas similar to those presented in [Ba94]. For example, Hochbaum and Maass [HM85, HM87] developed polynomial time approximation schemes for covering and packing problems in the plane. Feder and Greene [FG88] devised an approximation scheme for a geometric location problem related to clustering. Jiang and Wang [JW94] presented an approximation scheme for the Steiner tree problem in the plane when the given set of regular points is *c-local* (also called *civilized*). Recently, Eppstein [EP95] obtained efficient algorithms for the subgraph isomorphism problem for graphs of fixed genus.

In [MHR94, MR + 97, MH + 94] we investigated the existence and/or nonexistence of polynomial time approximations and approximation schemes for several PSPACE-hard problems for hierarchically specified instances. In [MH + 94b], we developed a general approach to prove PSPACE-hardness results for succinctly specified graphs. Condon *et al.* [CF + 93, CF + 94] characterized PSPACE in terms of probabilistically checkable debate systems and used this characterization to investigate the existence and nonexistence of polynomial time approximation algorithms for PSPACE-complete problems. In particular, they gave a polynomial time approximation algorithm for a maximization version of the QBF problem in which each clause has an existentially quantified variable. Further, they showed that unless P = PSPACE, it is not possible to obtain a polynomial time approximation scheme for this problem. They also gave PSPACE-hardness results concerning approximability of several other natural PSPACE-hard functions.

The remainder of this paper is organized as follows. In Section 3, we give some preliminary definitions. In Section 4, we discuss our approxima-

[2] Problems such as dominating set do not admit separator based approximation algorithms, even for planar graphs.

tion schemes for problems restricted to unit disk graphs. In Section 5, we discuss our ideas for graphs drawn in a civilized manner and extensions to λ-precision unit disk graphs. In Section 6, we extend the results in Sections 4 and 5 to obtain approximation schemes for problems restricted to unit disk graphs specified using k-near-consistent BOW-specifications. Section 7 briefly discusses extensions of our results, and Section 8 concludes the paper.

3. DEFINITIONS AND PRELIMINARIES

We have defined unit disk graphs as intersection graphs of unit disks in the plane. This model for unit disk graphs will be referred to as the *intersection model* [CCJ90]. As already mentioned, we assume that the disks are specified by the coordinates of their centers. The above definition of unit disk graphs can be extended so as to define intersection graphs of *regular polygons*. Let us call a p-sided polygon a *unit regular polygon* if the polygon is inscribed in a circle of radius 1 and the sides of the polygon are all equal. Each such polygon can be uniquely specified up to rotation by the number of sides and the coordinates of the center of the polygon. The above definitions can be easily extended to define intersection graphs of unit balls and unit regular polygons in higher dimensions. We now define graphs drawn in a civilized manner.

DEFINITION 3.1 [Te91]. For each pair of reals $r > 0$ and $s > 0$, a graph G can be drawn in R^d in an (r, s)-*civilized manner* if its vertices can be mapped to points in R^d so that

1. the length of each edge is $\leq r$, and
2. the distance between any two points is $\geq s$.

A civilized layout of a graph that can be drawn in a civilized manner in R^d consists of the coordinates of the vertices in R^d and the set of edges in the graph. We assume throughout this paper that the dimension (d) of the Euclidean space considered is at least 2. Graphs drawn in a civilized manner have been studied in the context of random walks by Doyle and Snell [DS84] and in the context of finite element analysis by Vavasis [Va91].

Define a *planar* (r, s)-*civilized graph* to be an (r, s)-civilized graph whose vertices can be embedded in the Euclidean plane (i.e., R^2). We discuss our algorithms for planar (r, s)-civilized graphs. But it will be clear that all the algorithms extend directly to civilized graphs drawn in higher dimensions albeit with slightly worse performance guarantee versus time trade-offs.

For the remainder of this section, we use "(r, s)-civilized graphs" to mean planar (r, s)-civilized graphs.

Next we define λ-precision unit disk graphs.

DEFINITION 3.2. For any fixed $\lambda > 0$, consider a finite set of unit disks in the plane where the centers of any two disks are at least λ apart.

A *λ-precision unit disk graph* $G(V, E)$ corresponding to the above set of unit disks is defined as follows: The vertices of G are in one-to-one correspondence with the set of unit disks and two vertices are joined by an edge iff the corresponding disks intersect.

Our definition of λ-precision unit disk graphs is motivated by the observation that practical problems, when modeled as problems on unit disk graphs, seldom have unit disk centers placed in a continuous fashion. For example, in VLSI designs, λ is a parameter determined by the fabrication process. It can be seen that grid graphs[3] are λ-precision unit disk graphs, for any $0 < \lambda \leq 2$. Also, each unit disk graph is a λ-precision unit disk graph for some $0 < \lambda \leq 2$. It is also easy to see that λ-precision unit disk graphs need not be planar.

We refer the reader to [GJ79, CLR91, Ba94] for definitions of the graph problems considered in this paper. We only recall the definition of treewidth bounded graphs here.

DEFINITION 3.3 (Treewidth bounded graphs). The class of *k-trees* [ALS91] is defined recursively as follows.

— A clique of size $k + 1$ is a k-tree.

— A k-tree with $n + 1$ vertices can be obtained from a k-tree with n vertices by adding a new vertex and edges from the new vertex to a set of k completely connected vertices.

A *partial k-tree* is a subgraph of a k-tree. The minimum value of k for which a graph is a subgraph of a k-tree is called the *treewidth* of the graph [ALS91].

Next, we recall a theorem of Bodlaender [Bo88], showing that for treewidth bounded graphs a number of optimization problems can be solved in polynomial time.

THEOREM 3.1. *For each fixed $k \geq 0$, given a graph of treewidth at most k, there is a linear time algorithm for solving the following problems: maximum independent set, minimum vertex cover, minimum edge dominating set, minimum dominating set, maximum cut, maximum triangle matching, and maximum H-matching.*

[3] A grid graph is a unit disk graph in which all the centers have coordinates that are even integers.

Since there are n node grid graphs with treewidth $\Theta(\sqrt{n})$, λ-precision unit disk graphs do not belong to the class of partial k-trees for any constant k. Therefore, techniques for partial k-trees cannot be directly applied to λ-precision unit disk graphs.

4. UNIT DISK GRAPHS

4.1. *Basic Technique*

As pointed out earlier, the *shifting strategy* was used by Baker [Ba94] for obtaining polynomial time approximation schemes (PTASs) for problems restricted to planar graphs, by Hochbaum and Maass [HM85, HM87] for devising PTASs for certain covering and packing problems in the plane, and by Feder and Greene [FG88] for obtaining a PTAS for a certain location problem. Consider a problem Π which can be solved by a divide-and-conquer approach with performance guarantee of ρ. The shifting strategy allows us to bound the error of the simple divide-and-conquer approach by applying it iteratively and choosing the best solution among these iterations as the solution to Π.

We outline the basic technique by discussing our NC-approximation scheme for the maximum independent set (MIS) problem for unit disk graphs. Given a set of n unit disks in the plane enclosed in an area I, we first divide the set into horizontal strips of width 2. Given an $\epsilon > 0$, we calculate the smallest integer k such that $(k/(k+1))^2 \geq 1 - \epsilon$. Next, for each i, $0 \leq i \leq k$, we partition the set of disks into l disjoint sets $G_1, \dots, G_l$ by removing disks in horizontal strips congruent to $i \bmod (k+1)$. Each strip is left closed and right open. A disk is said to lie in a given strip if its center lies in that strip. For each subgraph G_p, $1 \leq p \leq l$, we find an independent set of size at least $k/(k+1)$ times the size of an optimal independent set in G_p. The independent set for this partition is just the union of independent sets for each G_p. By an argument similar to the shifting lemma in [HM85], it follows that the iteration in which the partition yields the largest solution value contains at least $(k/(k+1))^2 \cdot$ $OPT(G)$ nodes, where $OPT(G)$ denotes the size of a maximum independent set in G. The algorithm runs in $n^{O(k^2)}$ time. As will be shown in Section 4.5, the algorithm can be implemented in NC. Other graph problems can also be solved similarly. In the case of minimization problems, instead of partitioning the set of unit disks, the subgraph G_l consists of the set of disks that lie between the $(l-1)$st horizontal strip congruent to $i \bmod (k+1)$ and the lth horizontal strip congruent to $i \bmod (k+1)$, including the disks in the horizontal strips.

4.2. *Approximation Scheme for the MIS Problem for Unit Disk Graphs*

Our algorithm (FMIS) for the MIS problem for unit disk graphs is given below.

4.3. *Finding an Optimal Solution in Step* 3(a)iiiD *of Algorithm FMIS*

We now discuss how to obtain an optimal solution for the independent set problem in Step 3(a)iiiD of the algorithm. By a simple packing argument, it can be shown that, for any fixed $k > 0$, the size of a maximum independent set of a unit disk graph, all of whose disks lie in a square of side k, is $O(k^2)$. This immediately gives us a way of finding an optimal solution in parallel. We can get a more efficient algorithm in the sequential case by considering the subgraph whose disks lie in a strip of width k (i.e., we do not have to execute the **For** loop for i_1). For each such unit disk graph, we can obtain an optimal independent set by means of dynamic programming. As the subsequent analysis will indicate, this gives an approximation algorithm with performance guarantee $k/(k+1)$.

ALGORITHM FMIS. *Input*—a set G of unit disks specified using coordinates of their centers.

```
1. Find the smallest integer $k$ such that $(k/(k+1))^2 \geq 1 - \epsilon$.
2. Divide the plane into horizontal strips of width 2.
3. Divide each horizontal strip into vertical strips of width 2.
   (a) For each $i$, $0 \leq i \leq k$, do
       (i) Partition the set of disks into $r$ disjoint sets $G_{i,1} \cdots G_{i,r}$ by removing all the disks in every horizontal strip congruent to $i \bmod (k+1)$.
       (ii) $G_i = \bigcup_{1 \leq j \leq r} G_{i,j}$.
       (iii) For each $j$, $1 \leq j \leq r$, do
             A. For each $i_1$, $0 \leq i_1 \leq k$, do
             B. Partition the set of disks in the set $G_{i,j}$ into $s_j$ disjoint sets $G_{i,j}^{i_1,1} \cdots G_{i,j}^{i_1,s_j}$ by removing every vertical strip congruent to $i_1 \bmod (k+1)$.
             C. $G_{i,j}^{i_1} = \bigcup_{1 \leq j_1 \leq s_j} G_{i,j}^{i_1,j_1}$.
             D. For each $G_{i,j}^{i_1,j_1}$, $1 \leq j_1 \leq s_j$, solve the problem optimally. Let the optimal value be denoted by $IS(G_{i,j}^{i_1,j_1})$.
             E. $IS(G_{i,j}^{i_1}) = \bigcup_{1 \leq j_1 \leq s_j} IS(G_{i,j}^{i_1,j_1})$.
       iv. $IS(G_{i,j}) = \max_{0 \leq i_1 \leq k} IS(G_{i,j}^{i_1})$.
   (b) $IS(G_i) = \bigcup_{1 \leq j \leq r} IS(G_{i,j})$.
4. $IS(G) = \max_{0 \leq i \leq k} IS(G_i)$.
```

Output—the set $IS(G)$ of independent unit disks.

4.4. *Performance Guarantee*

We next prove that the algorithm given above indeed computes a near optimal independent set. That is, given any $\epsilon > 0$, the algorithm will compute an independent set whose size is at least $(1 - \epsilon)$ times that of an optimal independent set.

We first prove that of all the different iterations for i (Step 3(a) of Algorithm FMIS), at least one iteration has the property that the number of nodes that are not considered in the independent set computation is a small fraction of an optimal independent set.

Recall that for each i we did not consider the explicit vertices in levels $j_1, j_2 \cdots j_{p_i}$, where $j_l = i \bmod (k+1)$, $1 \le l \le p$. For each i, $0 \le i \le k$, let S_i be the set of unit disks which were *not* considered in iteration i. Let $IS_{\text{opt}}(S_i)$ denote the vertices in the set S_i which were chosen in the optimal independent set $OPT(G)$.

LEMMA 4.1.

$$\max_{0 \le i \le k} |OPT(G_i)| \ge \frac{k}{(k+1)} |OPT(G)|.$$

Proof. The proof follows by observing that the following equations hold:

$$0 \le i, j \le k, i \ne j, \qquad S_i \cap S_j = \phi, \qquad \bigcup_{t=0}^{t=k} S_t = V(G)$$

since different levels are considered in different iterations. From the above set of equations, we have

$$|IS_{\text{opt}}(S_0)| + |IS_{\text{opt}}(S_1)| + \cdots + |IS_{\text{opt}}(S_k)| = |OPT(G)|.$$

Therefore,

$$\min_{0 \le t \le k} |IS_{\text{opt}}(S_t)| \le \frac{|OPT(G)|}{(k+1)},$$

$$\max_{0 \le i \le k} |OPT(G_i)| \ge |OPT(G)| - \min_{0 \le t \le k} |IS_{\text{opt}}(S_t)|$$

$$\ge \frac{k}{(k+1)} |OPT(G)|. \quad \blacksquare$$

THEOREM 4.2. $|IS(G_{i,j})| \ge (k/(k+1)) \cdot |OPT(G_{i,j})|$.

Proof. Fix an iteration i and consider each of the individual graphs $G_{i,j}$. Applying Lemma 4.1 to the unit disk graph $G_{i,j}$ we get that for some $0 \le i_1 \le k$

$$\left|OPT\left(G_{i,j}^{i_1}\right)\right| \ge \frac{k}{(k+1)}\left|OPT\left(G_{i,j}\right)\right|.$$

Now, by Step 3(a)iii of the algorithm we have

$$\begin{aligned}\left|OPT\left(G_{i,j}^{i_1}\right)\right| &= \sum_{j_1=1}^{j_1=s_j}\left|OPT\left(G_{i,j}^{i_1,j_1}\right)\right|,\\ \left|OPT\left(G_{i,j}\right)\right| &= \sum_{j=1}^{j=r}\left|OPT\left(G_{i,j}\right)\right|.\end{aligned} \tag{1}$$

Using the above equations we get

$$\begin{aligned}\left|IS\left(G_{i,j}\right)\right| &= \max_{0\le i_1\le k}\left|IS\left(G_{i,j}^{i_1}\right)\right| && \text{(by Step 4)}\\ &= \max_{0\le i_1\le k}\sum_{j_1=1}^{j_1=s_j}\left|IS\left(G_{i,j}^{i_1,j_1}\right)\right| && \text{(by Step 3(b))}\\ &= \max_{0\le i_1\le k}\sum_{j_1=1}^{j_1=s_j}\left|OPT\left(G_{i,j}^{i_1,j_1}\right)\right| && \text{(by Step 3(a)iiiD)}\\ &= \max_{0\le i_1\le k}\left|OPT\left(G_{i,j}^{i_1}\right)\right| && \text{(by Eq. (1))}\\ &\ge \left(\frac{k}{k+1}\right)\cdot\left|OPT\left(G_{i,j}\right)\right| && \text{(by Lemma 4.1)}. \quad \blacksquare\end{aligned}$$

By a repeated application of the above lemma we get

THEOREM 4.3. $|IS(G)| \ge (k/(k+1))^2 \cdot |OPT(G)|$.

Proof. We consider the iteration when the value of i is such that $|OPT(G_i)| \ge (k/(k+1))|OPT(G)|$. By Lemma 4.1 such an i exists. Also note that

$$\left|OPT(G_i)\right| = \sum_{j=1}^{j=r}\left|OPT\left(G_{i,j}\right)\right|.$$

Using the above equations we get that

$$
\begin{aligned}
|IS(G)| &= \max_{0 \le i \le k} |IS(G_i)| \\
&= \max_{0 \le i \le k} \sum_{j=1}^{j=r} |IS(G_{i,j})| \qquad \text{(by Step 3(b))} \\
&\ge \frac{k}{k+1} \max_{0 \le i \le k} \sum_{j=1}^{j=r} |OPT(G_{i,j})| \qquad \text{(by Theorem 4.2)} \\
&\ge \frac{k}{k+1} \max_{0 \le i \le k} |OPT(G_i)| \qquad \text{(by Step 3(b))} \\
&\ge \left(\frac{k}{k+1}\right)^2 \cdot |OPT(G)| \qquad \text{(by Lemma 4.1)}. \quad \blacksquare
\end{aligned}
$$

4.5. *Running Time*

We now estimate the running time of Algorithm FMIS. As mentioned earlier, the size of a maximum independent set in each square is no more than $O(k^2)$. The loop in Step 3(a) is executed for $k + 1$ different values of i. For each value of i the running time in Step 3(a)iii is $n^{O(k^2)}$. Hence the total running time (work) of Algorithm FMIS is $n^{O(k^2)}$.

4.6. *Better Time and Performance in the Sequential Case*

We can get a better time and performance by solving the MIS problem for each graph $G_{i,j}$ optimally using dynamic programming. The algorithm is the same as the previous algorithm, with the only difference that Step 3(a)iii in the previous algorithm is replaced by the step to find an optimal independent set in each $G_{i,j}$. We now describe the idea behind the dynamic programming algorithm. Consider the unit disks whose centers lie in a rectangular slice RS of height $2k$ and width 2 in $G_{i,j}$. Since $G_{i,j}$ is a unit disk graph, RS can contain no more than $3k + 3$ mutually nonintersecting unit disks. This gives us a bound on the size of a maximum independent set of a set of unit disks whose centers lie in RS. Furthermore, removal of the unit disks in RS breaks the set of unit disks in $G_{i,j}$ into disjoint sets L and R. Let ρ be the number of unit disks in R. For each combination of at most $3k + 3$ nodes in RS, we get a possible maximum independent set of the unit disks in RS. For the two subgraphs $RS \cup L$ and $RS \cup R$ we keep a table with no more than $O(\rho^{3k+3})$ entries of the maximum independent set for each subgraph for each possible maximum independent set in RS. The union of maximum independent sets for the two subgraphs when they agree on the nodes selected from RS gives us an independent set for the whole graph. The running time of the dynamic programming procedure is $n^{O(k)}$. Therefore, the total running time of the algorithm is $n^{O(k)}$. The approximation algorithm has a performance guarantee of $k/(k+1)$.

4.7. *Other Optimization Problems*

We now discuss how these ideas can be applied to other combinatorial problems. Since many of the ideas in this subsection are also discussed in [Ba94], our discussion is brief.

(1). *Minimum Vertex Cover*

In order to approximate the Minimum Vertex Cover problem for unit disk graphs, we consider overlapping pieces of unit disk graphs. We discuss the sequential approximation algorithm providing a vertex cover of size no more than $(k+1)/k$ times that of an optimal vertex cover. The idea is identical to the one discussed in [Ba94]. Consider a geometric representation of a unit disk graph G. In Step 3(a)i of Algorithm FMIS, the unit disk graph $G_{i,j}$, $0 \le j \le r$, is obtained by considering the set of unit disks belonging to horizontal strips $jk+i$ to $(j+1)k+i$. We can find a good vertex cover for each of the subgraphs, by observing that if IS is a maximum independent set in a graph $G(V,E)$, then $V - IS$ is a minimum vertex cover. For each i, the union over all j of the vertex cover for each graph $G_{i,j}$, yields a valid vertex cover for G. The algorithm picks the best among all the vertex covers obtained for the different values of i. Using $OPT(G)$ to denote an optimal vertex cover for G, we have:

LEMMA 4.4. *The size of the vertex cover obtained is no more than* $((k+1)/k)|OPT(G)|$.

Proof. Fix an optimal solution $OPT(G)$ to the vertex cover problem. Then, for some $0 \le t < k$, at most $|OPT(G)|/k$ nodes in $OPT(G)$ are in horizontal strips congruent to $t \pmod{k}$. Consider the iteration when the unit disk graphs are obtained by overlapping at horizontal strips congruent to $t \pmod{k}$. The size of the vertex cover obtained in this iteration is no more than $|OPT(G)| + |OPT(G)|/k$, since the overlapping horizontal strips are counted twice. Since the heuristic picks the minimum vertex cover over all values of i, it follows that the size of the vertex cover produced by the heuristic is no more than $((k+1)/k)|OPT(G)|$. ▮

In order to obtain a parallel approximation scheme, we just need to obtain overlapping unit disk graphs for each set of horizontal strips to obtain $G_{i,j}^{i_1,j_1}$ in Step 3(a)iiiB. This yields a vertex cover of size at most $((k+1)/2)^2|OPT(G)|$.

(2). *Minimum Dominating Set*

The basic idea is similar to the minimum vertex cover problem. However, there is a subtle difference between our approximation schemes for the vertex cover and dominating set problems. To see why, let G be the given unit disk graph and let k be the fixed integer determined by the performance guarantee parameter ϵ. Suppose we partition the nodes of G into strips as in Algorithm FMIS and consider the subgraph G' induced by the vertices in k consecutive strips, say $i, i+1, \dots, i+k-1$. Let $OPT_{VC}(G)$ and $OPT_{DS}(G)$ denote an optimal vertex cover and an optimal dominating set for G, respectively. It can be seen that the projection of $OPT_{VC}(G)$ on to the chosen set of k consecutive strips is a vertex cover for G'. However, the projection of $OPT_{DS}(G)$ on to the k consecutive strips need not be a dominating set for G'; an optimal dominating set for G may choose to dominate some or all the vertices in G' by using vertices from strip $i-1$ or from strip $i+k$. Thus, an approximation scheme for the minimum dominating set problem must explicitly account for this difference.

Consider the geometric representation of a unit disk graph G. The outline of our approximation scheme for the dominating set problem is very similar to Algorithm FMIS except for the following.

(a) In Step 1, we find the smallest integer k such that $((k+1)/k)^2 \leq 1+\epsilon$.

(b) In Step 3(a)iiiD, an optimal dominating set for the subgraph formed by the unit disks in a $k \times k$ square Q is computed as follows. We expand Q by one unit in each direction; that is, we consider the $(k+1) \times (k+1)$ square Q' surrounding Q. The vertices to be dominated are those in Q. However, in doing so, we may use the vertices in Q'.

Finding an optimal dominating set for Q, possibly containing one or more vertices from the region $Q' - Q$, can be done in polynomial time as follows. It is easy to see that any maximal independent set in a graph is also a dominating set. Every independent set in Q' is of size at most $(k+1)^2$. So, by considering all subsets of size at most $(k+1)^2$ from Q', we can obtain an optimal dominating set for the vertices in Q. Since the number of such subsets is $n^{O(k^3)}$ and k is fixed, this procedure runs in polynomial time.

A proof that the resulting algorithm produces a dominating set whose size is at most $((k+1)/k)^2$ that of an optimal dominating set can be given along lines similar to that for the vertex cover problem above.

The results discussed in this section are summarized in the following theorem.

THEOREM 4.5. *For unit disk graphs, there are NC-approximation schemes for the following problems: maximum independent set, minimum vertex cover, and minimum dominating set.*

5. GRAPHS DRAWN IN A CIVILIZED MANNER

Our ideas in the previous section can be applied to problems for graphs drawn in a civilized manner. The approximation schemes assume that a civilized layout of the graph is given. For such graphs, we observe that each small subgraph obtained as a result of decomposition has a small treewidth and use this observation to devise approximation schemes with a better performance guarantee versus time trade-off.

5.1. *MIS Problem for Civilized Graphs*

The algorithm for solving the MIS problem for civilized graphs is very similar to Algorithm FMIS. Consequently, we only point out the differences here.

1. In Step 2, we divide the plane into horizontal strips of width r. Each point, which represents a node of the given (r, s)-civilized graph, can now be assigned to a strip. (When a point lies on the boundary between two successive strips, it is assigned to the bottom strip.) Since the given graph is (r, s)-civilized, this method of partitioning points into strips ensures that removal of all the points in a strip disconnects the underlying graph.

2. Step 3(a)iii in the Algorithm FMIS is replaced by the step to find an optimal independent set in each $G_{i,j}$. (This can be done in polynomial time since the treewidth of $G_{i,j}$ is bounded as indicated in the next theorem.)

THEOREM 5.1. *Consider a civilized graph $G_{i,j}$ obtained in each iteration i of Step* 3(a)i *in Algorithm FMIS. The treewidth of $G_{i,j}$ is $O(k)$.*

Proof. As in the dynamic programming formulation for unit disk graphs, consider the vertices in a rectangular slice of side height rk and width r in $G_{i,j}$. Since $G_{i,j}$ is an (r, s)-civilized graph, the maximum number of vertices in this square region is at most $k \cdot (r^2/s^2)$. Furthermore, removal of the vertices in this square breaks the graph into disjoint pieces. By recursively applying the above idea on each smaller piece, we can construct a tree decomposition of the graph $G_{i,j}$ with treewidth $k \cdot (r^2/s^2)$. Since r and s are fixed, the treewidth is $O(k)$. ▮

Given Theorem 5.1, we can use efficient dynamic programming algorithms for solving problems for treewidth bounded graphs as summarized in Theorem 3.1 to obtain approximation schemes with better performance guarantee versus running time trade-off. We now establish the performance guarantee of our approximation algorithm.

THEOREM 5.2. *For all fixed $r, s \geq 0$, given a graph drawn in an (r,s)-civilized manner, there is a linear time approximation scheme for the MIS problem.*

Proof. As in the case of unit disk graphs, we have

$$|OPT(G_i)| = \sum_{j=1}^{j=r} |OPT(G_{i,j})|$$

and

$$\begin{aligned} |IS(G)| &= \max_{0 \leq i \leq k} |IS(G_i)| \\ &\geq \max_{0 \leq i \leq k} |OPT(G_i)| \qquad \text{(by new Step 3(b)iii)} \\ &\geq \left(\frac{k}{k+1}\right) \cdot |OPT(G)| \qquad \text{(by Lemma 4.1)}. \end{aligned}$$ ▮

Note that, for fixed r and s, (r,s)-civilized graphs have linear time (more precisely, $O(2^k n)$ time) approximation schemes for the problems considered here since these problems can be solved in $O(2^k n)$ time for graphs of treewidth k [AP89]. In contrast, for arbitrary unit disk graphs, our approximation schemes (as discussed in Section 4.6) have a running time of $n^{O(k^2)}$.

5.2. *Other Problems for Civilized Graphs*

Similar approximation schemes can be derived for various other optimization problems for (r,s)-civilized graphs. We omit the details since they are very similar to that of the MIS problem. We have the following theorem.

THEOREM 5.3. *For all fixed $r, s \geq 0$, given a graph drawn in an (r,s)-civilized manner, there are polynomial time approximation schemes for solving the following problems: maximum independent set, minimum vertex cover, minimum edge dominating set, minimum dominating set, maximum cut, maximum triangle matching, and maximum H-matching.*

It can be seen that, for any $\lambda > 0$, a λ-precision unit disk graph can be drawn in a $(2, \lambda)$-civilized manner. Thus, Theorem 5.3 also yields approximation schemes for a number of problems for λ-precision unit disk graphs.

It is easy to see that our results discussed in the previous section can be extended to geometric intersection graphs of other regular polygons including intersection graphs of isothetic unit squares and also to intersection graphs of regular geometric objects in higher dimensions. In each of these cases, the running time and the performance guarantee of the algorithm depend on the geometric objects considered.

6. BOW-SPECIFIED GEOMETRIC INTERSECTION GRAPHS

Next, we discuss our ideas for obtaining approximation algorithms for the MIS problem for a set of unit disks specified using BOW-specifications. The basic approach for obtaining approximation schemes is similar to the one given in [MH + 94], and thus we keep the discussion brief.

6.1. *BOW-specifications*

The specification language used here to describe a set of unit disks hierarchically is almost identical to the BOW-specification language used to describe a set of isothetic rectangles. The only difference is that instead of the BOX command we have the DISK command whose syntax is as follows:

$$\mathsf{DISK}(x, y, r)$$

where (x, y) is the center of the disk and r is its radius. A *symbol* (also referred to as a *nonterminal*) in this language represents a collection of unit disks and has a unique identifier (symbol number) which is a positive integer. The description for a symbol consists of zero or more DISK commands and DRAW commands. The syntax of the DRAW command is as follows:

$$\mathsf{DRAW}\ \text{symbol\#}\ \text{at}\ (x, y)$$

Here the symbol# is the identifier of a *previously defined* symbol and (x, y) specifies the amount of translation to be applied to the centers of the disks defined in the specified symbol.

A BOW-specification $\Gamma = (G_1, \ldots, G_n)$ consists of a sequence of symbol definitions G_i, $1 \leq i \leq n$. Let G_i have n_i DRAW and DISK commands. Then the size of Γ, denoted by N, is given by $N = \sum_{1 \leq i \leq n} n_i$. The set of disks specified by Γ is the one corresponding to the symbol with the largest identifier (i.e., G_n).

EXAMPLE 1. Consider the following description of a set of unit disks described using a BOW-specification $\Gamma = (G_1, G_2, G_3)$.

1. DEFINE G_1
   DISK$(0, 0, 1)$
2. DEFINE G_2
   DRAW G_1 at $(0, 0)$
   DRAW G_1 at $(2, 0)$
   DRAW G_1 at $(0, 2)$
3. DEFINE G_3
   DRAW G_2 at $(0, 0)$
   DRAW G_2 at $(0, 4)$
   DRAW G_2 at $(4, 0)$

Note. Figure 1 shows the set of unit disks obtained by expanding the above BOW-specification.

With the set of unit disks defined as above, we associate an *intersection graph* which has one vertex per unit disk and two vertices joined by an edge if and only if the corresponding disks intersect. Graphs obtained by expanding the BOW-specifications of a set of unit disks will be referred to as BOW-*specified unit disk graphs*.

In [BOW83], Bentley *et al.* show that the general BOW-specifications are too powerful and make most of the natural problems intractable. They then define the notion of *consistent* BOW-*specifications* which are realized by adding an attribute called the MBR (minimum bounding rectangle) to each symbol [BOW83]. This attribute denotes the smallest bounding rect-

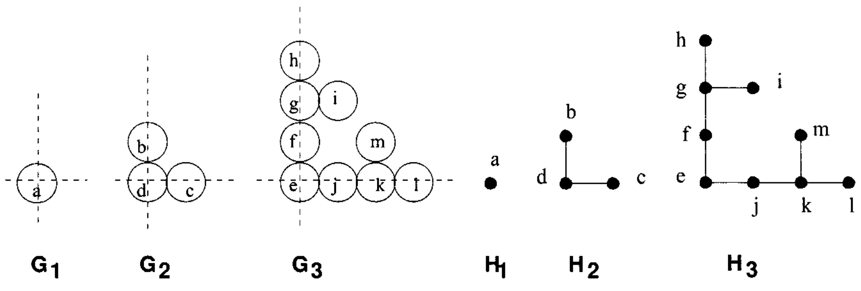

FIG. 1. A set of unit disks and its corresponding unit disk graph.

angle enclosing the set of unit disks associated with the symbol. The syntax of a symbol definition is as follows:

DEFINE symbol#

attribute: MBR([x-value], [y-value])([width], [height])

⟨Sequence of DRAW and DISK commands⟩

Here the pair ([x-value], [y-value]) denotes the coordinates of the bottom left corner of the minimum bounding rectangle (MBR), and the pair ([width], [height]) denotes the width and the height of MBR. So for instance, the following is a valid definition of a symbol G_1:

DEFINE G_1

attribute: MBR$(-1, -1)(4, 4)$

DISK$(0, 0, 1)$

DISK$(0, 2, 1)$

DISK$(2, 0, 1)$

DISK$(2, 2, 1)$

DEFINITION 6.1 [BOW83]. A BOW-specification is *consistent* if, for each symbol, the MBR of symbols called within the symbol and the disks explicitly defined in the symbol do not intersect.

As observed in [BOW83], consistency is a very strong restriction as any set of rectangles containing an intersecting pair cannot be represented using a consistent BOW-specification. Extending the definition of consistency, we define the concept of k-near-consistent BOW-specifications. We first need some additional notation. Associated with each BOW-specification $\Gamma = (G_1, \dots, G_n)$ (where G_n denotes the largest symbol) is a tree structure depicting the sequence of calls made by the symbols defined in Γ. Following Lengauer *et al.* [LW87a], we call it the *hierarchy tree* $HT(\Gamma)$ associated with Γ. Intuitively, near-consistent BOW-specifications allow one to have intersections which do not occur between explicit symbols defined too far away from each other in the hierarchy tree of the specification. Given a hierarchy tree $HT(\Gamma)$, we can associate a level number with each node (a symbol) in the tree. The *level number* of a node in $HT(\Gamma)$ is the number of edges in the path from the node to the root of the tree $HT(\Gamma)$.

We let $E(G_i)$ denote the set of unit disks obtained by expanding the hierarchy tree $HT(G_i)$. Thus $E(G_n)$ denotes the set of unit disks described by a given specification $\Gamma = (G_1, \dots, G_n)$.

DEFINITION 6.2. A BOW-specification is 1-near-consistent if and only if the following conditions hold:

1. For each symbol G_i, the MBR of G_i contains the MBR of all the symbols called in G_i.

2. The MBR of the symbols called in G_i do not intersect one another.

3. For each explicit disk u defined in G_i, u does not intersect the MBR of any symbol G_j such that G_j occurs in $HT(G_i)$ and level number of G_j in $HT(G_i)$ is ≥ 2.

The above definition can be easily extended to define k-near-consistent BOW-specifications, for any fixed $k \geq 1$. An example of 1-near-consistent BOW-specification of unit disks is given in Figure 2. Note that this is a strict extension of consistent BOW-specifications. With the above syntactic restriction, a natural question to ask is the following: Given a BOW-specification, how hard is it to verify that the specification obeys the above restriction? Our next theorem points out that the verification can be done in polynomial time.

THEOREM 6.1. *For any fixed $k \geq 1$, there is a polynomial time algorithm to determine if a given a BOW-specification $\Gamma = (G_1, \ldots, G_n)$, is k-near-consistent.*

Proof. We discuss our algorithm for checking if a BOW-specification is 1-near-consistent. The extension to check if the specification is k-near-consistent for any fixed k is straightforward. The algorithm proceeds in a bottom-up manner, processing one symbol at a time. When processing symbol G_i, it first verifies that the MBR of G_i contains the MBR of the symbols called in the definition of G_i. This verifies condition 1 in the

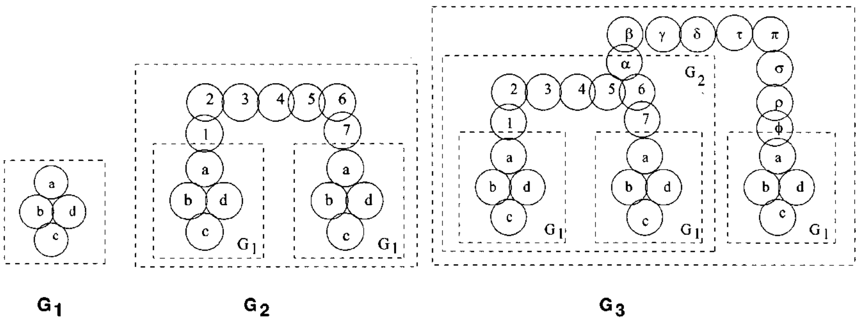

FIG. 2. A 1-near-consistent BOW-specification of a set of unit disks.

definition. It then verifies that the MBR of each of the symbols called in G_i are mutually nonintersecting. These checks can be performed in polynomial time using a polynomial time routine for finding intersecting rectangles. This verifies Condition 2 in the definition of near-consistent specification. Next, we process one explicit disk $u \in G_i$ at a time as follows. Let G_i call symbols $G_{j_1}, \ldots, G_{j_t}$. Furthermore, let the symbol G_{j_p}, $1 \leq p \leq t$, call symbols $G_1^{j_p}, \ldots, G_{r_p}^{j_p}$. Then we simply check that the unit disk $u \in G_i$ does not intersect the MBR of any of the symbols $G_{r_p}^{j_p}$, $1 \leq p \leq t$. This check can be performed in $O(N^2)$ time for each disk as the total number of symbols in the description is $O(N)$. It is easy to see that the dominant part of the running time is due to this check. Therefore, the total time taken to check whether the specification is 1-near-consistent is $O(N^3)$. ▌

6.2. *Meaning of Approximation Algorithms for BOW-Specified Problems*

Before we give details of our algorithms, it is important to understand what we mean by a *polynomial time approximation algorithm* for a problem Π, when the instance is specified using BOW-specifications. Corresponding to each decision problem Π, specified using BOW-specifications, we consider four variants of the corresponding optimization problem. We illustrate this with an example (see also [MHR94, MH + 94]).

EXAMPLE. Consider the minimum vertex cover problem, where the input is a succinct specification of a graph G and the goal is to compute the size of a minimum vertex cover for G. This will be referred to as the *size problem*. Our polynomial time approximation algorithm for the size problem computes the size of an approximate vertex cover and runs in time *polynomial* in the *size* of the succinct description, rather than the *size* of G. Moreover, it also solves in polynomial time (in the size of the succinct specification) the following *query problem*: Given any vertex v of G and its position in the expanded specification, determine whether v belongs to the approximate vertex cover so computed. Moreover, for all the problems considered here, we can construct in polynomial time a succinct specification of the approximate set computed (referred to as the *construction problem*). Finally, we can also solve the *output problem*; that is, output the approximate vertex cover in time that is linear in the size of the solution but uses space which is only polynomial in the size of the BOW-specification.

This is a natural extension of the definition of approximation algorithms, for problems specified using nonhierarchical (standard) specifications, since the number of vertices is polynomial in the size of the nonhierarchical description and hence a polynomial time approximation algorithm can solve the optimization, query, and output problems (approximately) in polynomial time.

6.3. *The Basic Technique*

We now give the basic technique behind all our approximation algorithms for unit disk graphs specified using k-near-consistent BOW-specifications. For the sake of simplicity, we assume that the BOW-specification is 1-near-consistent. Given a maximization problem Π in Table I (see Section 8), our approximation algorithm takes time $O(N \cdot T(N^{l+1}))$ to achieve a performance guarantee of $(l/(l+1)) \cdot FBEST$. Here, l is a constant that depends only on the performance guarantee parameter ϵ, $T(N^{l+1})$ denotes the running time of a heuristic which can process flat specifications of size $O(N^{l+1})$ and which has a performance guarantee $FBEST$. During an iteration i we delete all the explicit vertices which belong to nonterminals defined at level j, where $j = i \bmod(l+1)$. (For minimization problems, instead of deleting the vertices in the level, we consider the vertices as a part of both the subtrees.) This breaks up the given hierarchy tree into a collection of disjoint trees. The heuristic simply finds a near optimal solution for the vertex induced subgraph defined by each small tree and then outputs the union of all these solutions as the solution for the problem Π. (For a fixed l, the size of each subgraph is polynomial in the size of the specification.) It is important to observe that the hierarchy tree can have an exponential number of nodes and hence the deletion of nonterminals and finding near optimal solutions for each subtree must be done in such a manner that the whole process takes only polynomial time. This is achieved by observing that the subtrees can be divided into n distinct equivalence classes and that it is easy to count the number of subtrees in each equivalence class.

6.4. *Approximation Scheme for the MIS Problem*

We illustrate our ideas by giving a polynomial time approximation scheme for the maximum independent set (MIS) problem for unit disk graphs specified using a 1-near-consistent BOW-specification.

In the following description, we use $HIS(G_i)$ to denote the approximate independent set produced by the algorithm for the graph $E(G_i)$. Before we discuss the details of the heuristic, we define the concept of *partial expansion* of a 1-near-consistent BOW-specification. We note that, for each nonterminal G_i, there is a unique hierarchy tree $HT(G_i)$ rooted at G_i.

DEFINITION 6.3. The partial expansion $PE(G_i^j)$ of the graph associated with G_i of the BOW-specification Γ is constructed as follows:

$j = 0$: $PE(G_i^j) = G_i - \{u\colon u$ is an explicit unit disk defined in $G_i\}$. (Thus, the definition of $PE(G_i^0)$ now consists of the collection of symbols called in the definition of G_i.)

$j \geq 1$: Repeat the following steps for each symbol G_r called by G_i:

1. Compute the partial expansion $PE(G_r^{j-1})$ of G_r.
2. The coordinates of an explicit disk or a symbol in $PE(G_r^{j-1})$ are given by its relative position with respect to the MBR of G_r plus the offset of MBR of G_r.

(Observe that the definition of $PE(G_i^j)$ consists of (i) explicit unit disks defined in all the symbols at depth r, $0 \leq r \leq j-1$, in $HT(G_i)$ and (ii) a disjoint collection of symbols G_k, such that the symbol G_k occurs at depth $j+1$ in the hierarchy tree $HT(G_i)$.)

Let $Ex(PE(G_i^j))$ denote the subgraph induced by the set of explicit disks (vertices) in the definition of $PE(G_i^j)$. Also let $V(E(G_i))$ denote the set of vertices in $E(G_i)$. Our algorithm (HMIS) for computing a near-optimal independent set is given below.

ALGORITHM HMIS. *Input*—a 1-near-consistent BOW-specification $\Gamma = (G_1, \ldots, G_n)$ of a set of unit disks G and an integer l.

1. **For** each i, $1 \leq i \leq l$, find a near optimal independent set in $E(G_i)$ using Algorithm FMIS.
2. **For** each i, $l+1 \leq i \leq n-1$
   (a) Compute the partial expansion $PE(G_i^l)$ of G_i.
   (b) Find a near optimal independent set in the subgraph $Ex(PE(G_i^l))$ (the subgraph induced by the explicit vertices in the definition of $PE(G_i^l)$) using Algorithm FMIS. Denote this by A_i^l.
   (c) Let $G_{i_1}, \ldots, G_{i_p}$ denote the nonterminals called in $PE(G_i^l)$. Then the independent set for the whole graph for the iteration i, denoted by $HIS(G_i)$, is given by

$$HIS(G_i) = A_i^l \cup \bigcup_{1 \leq r \leq p} HIS(G_{i_r}).$$

Remark. The explicit vertices in $PE(G_i^l)$ do not have an edge to any of the nonterminals $G_{i_1}, \ldots, G_{i_p}$. From this observation and the definition of

BOW-specification it follows that

$$|HIS(G_i)| = |A_i^l| + \sum_{1 \le r \le p} |HIS(G_{i_r})|.$$

3. **For** each $0 \le i \le l$

   (a) Obtain the partial expansion of $PE(G_n^i)$ of G_n.

   (b) Find a near optimal independent set of all the explicit vertices in $PE(G_n^i)$ using Algorithm FMIS. Denote this by A_n^i.

   (c) Let $G_{n_1}, \ldots, G_{n_p}$ denote the nonterminals called in $PE(G_n^i)$. The independent set for the whole graph for the iteration i, denoted by $HIS_i(G_n)$, is computed using the equation

$$HIS_i(G_n) = A_n^i \cup \bigcup_{1 \le r \le p} HIS(G_{n_r}).$$

Remark. By a remark similar to the one following Step 2(c) above, we have

$$|HIS_i(G_n)| = |A_n^i| + \sum_{1 \le r \le p} |HIS(G_{n_r})|.$$

4. $HIS(G) = \max_{0 \le i \le l} HIS_i(G_n)$.

Output—a BOW-specification of an independent set whose size is at least $(l/(l+1))^2$ times the size of an optimal independent set.

6.5. *Performance Guarantee and Running Time*

The correctness of the above algorithm follows from Lemmas 6.2–6.5. The lemmas can be proven by induction on the number of nonterminals in the definition of Γ and are a consequence of the definition of a partial expansion and level-restrictedness of the given specification.

LEMMA 6.2. *Consider the graph $E(G_i)$ corresponding to the nonterminal G_i. In each iteration i, $l+1 \le i \le n-1$, Step 2 of Algorithm HMIS computes an independent set in the graph induced by the vertices $V(E(G_i)) - \mathscr{V}_i$. Here $\mathscr{V}_i$ denotes the set of explicit vertices defined in nonterminals at levels $j = i \bmod (l+1)$ in the hierarchy tree $HT(G_i)$.*

Proof. Induction on the depth of the hierarchy tree associated with G_i.

Basis. If the depth $\le l$, the proof follows directly.

Induction. Assume that the lemma holds for all hierarchy trees of depth up to m. Consider a hierarchy tree of depth $m+1$. Step 2(c) of the algorithm computes a partial expansion of $PE(G_i^l)$. This implies that all the explicit vertices at level l in the hierarchy tree $HT(G_i)$ were deleted. Each of the nonterminals left in $PE(G_i^l)$ is at level $l + 1$ in $HT(G_i)$. Now, each of the nonterminals G_t left in $PE(G_n^l)$ has an associated hierarchy tree of depth $\leq m$. The proof then follows by induction. ▮

LEMMA 6.3. *In each iteration i of Step 3 of Algorithm HMIS, all the explicit vertices defined in nonterminals at levels $j = i \bmod(l+1)$ in the hierarchy tree $HT(G_n)$ are effectively deleted.*

Proof. Consider a hierarchy tree $HT(G_n)$. In Step 3 of Algorithm HMIS we compute a partial expansion for the first i levels. The partial expansion removes all the explicit vertices defined in nonterminals at level i. Also, by the definition of partial expansion it follows that all explicit vertices defined in nonterminals at levels 1 to i appear explicitly in the partially expanded graph. Therefore, the partially expanded graph now consists of all nonterminals defined at level $i + 1$ in the hierarchy tree $HT(G_n)$. The theorem now follows as a consequence of Lemma 6.2. ▮

LEMMA 6.4. *For a given iteration i, the removal of explicit vertices at levels $j = i \bmod(l+1)$ decomposes the given hierarchical graph $E(\Gamma_n)$ into a collection of disjoint subgraphs.*

Proof. By the definition of 1-near-consistent specification it follows that there is no edge between a copy of a nonterminal defined at level m and one defined at level $m + 2$. ▮

Given the decomposition of the hierarchical graph into a collection of vertex disjoint subgraphs, we can associate a hierarchy subtree with each of the subgraphs. Each such subtree can be labeled by the type of nonterminal which is the root of the subtree. Since the hierarchical specification is a restricted form of a context-free grammar producing a single word, it can be easily seen that, during any iteration i, the number of distinct labels used to label the subtrees is less than n. Hence we have the following lemma.

LEMMA 6.5. *For each iteration i of Step 3 of Algorithm HMIS, the root of each subtree is labeled by one of the elements of the set $\{G_1, \dots, G_{n-1}\}$.*

LEMMA 6.6. *For each $1 \leq i \leq n$, let $H_1^i, H_2^i, \dots, H_{r_i}^i$ be the set of graphs corresponding to the subtrees with roots labeled G_i. Then $H_1^i, H_2^i, \dots, H_{r_i}^i$ are isomorphic.*

Proof. Follows from the definition of partial expansion of a nonterminal. ▮

Let $IS(T)$ denote the size of a maximum independent set of the graph corresponding to the subtree T of the hierarchy tree obtained during Step 3 of the algorithm. For a given iteration i, let $IS(F_i) = \bigcup_{T \in F_i} IS(T)$, where the union is taken over all the trees in the forest F_i obtained during iteration i as a result of decomposition. By Lemma 6.3 it follows that for each iteration i we did not consider the explicit vertices in levels $j_1, j_2, \ldots, j_{p_i}$, where $1 \le q \le p_i$ and $j_q = i \bmod (l+1)$. Let $IS(G)$ denote a maximum independent set for G. The next lemma points out that at least one iteration of Step 3 has the property that the number of nodes that are *not* considered in the independent set computation is a small fraction of the optimal independent set. The proof of the lemma is omitted as it is similar to that of Lemma 4.1.

LEMMA 6.7.

$$\max_{0 \le i \le l} |IS(F_i)| \ge \frac{l}{(l+1)} |IS(G)|. \quad \blacksquare$$

We now argue that Algorithm HMIS generates a valid independent set. To see this, note that, for each $1 \le i \le n-1$, we compute an independent set in Steps 1 and 2. This follows from the correctness of Algorithm FMIS. Next consider each iteration of Step 3. Step 3(b) computes an independent set for the explicit vertices $Ex(PE(G_n^i))$. Step 3(c) combines the independent sets obtained in Step 3(b) and the independent sets in the graphs $E(G_{n_r})$, $1 \le r \le p$. By noting that Γ is a 1-near-consistent specification, it follows that the sets merged in Step 3(c) are disjoint; thus $HIS_i(G_n)$ is an independent set.

We now prove that Algorithm HMIS has the claimed performance guarantee.

LEMMA 6.8. $HIS_i(G_n)| \ge (l/(l+1)) \cdot |IS(F_i)|$.

Proof. Induction on the number of nonterminals in the definition of T. The base case is straightforward. Consider the induction step. By the definition of partial expansion it follows that

$$|IS(F_i)| = \left|IS\left(Ex\left(PE\left(G_n^i\right)\right)\right)\right| + \sum_{1 \le r \le p} \left|IS\left(PE\left(G_{n_r}\right)\right)\right|.$$

From Step 3(c) of the algorithm HMAX-IS we also know that

$$|HIS_i(G_n)| = |A_n^i| + \sum_{1 \le r \le p} \left|HIS\left(G_{n_r}\right)\right|.$$

Using the induction hypothesis and Theorem 4.3 it follows that

$$|A_n^i| \geq \left(\frac{l}{l+1}\right) \cdot \left| IS\big(Ex\big(PE\big(G_n^i\big)\big)\big) \right|$$

and

$$\left| HIS\big(G_{n_r}\big) \right| \geq \left(\frac{l}{l+1}\right) \cdot \left| IS\big(PE\big(G_{n_r}\big)\big) \right|.$$

The lemma now follows. ∎

THEOREM 6.9. $HIS(G) \geq (l/(l+1))^2 \cdot |IS(G)|$.

Proof. Follows from Lemmas 6.7 and 6.8. ∎

We now analyze the running time of Algorithm HMIS.

THEOREM 6.10. *For any fixed $k \geq 0$ and $l \geq 1$, given a k-near-consistent BOW-specification Γ of a set of unit disks, Algorithm HMIS runs in time $N^{O(l^2)}$ and computes an independent set of size at least $(l/(l+1))^2$ times the size of an optimal independent set, where N is the size of Γ.*

Proof. Consider Step 1. The number of explicit vertices in $E(G_i)$, $1 \leq i \leq l$, is $O(N^l)$. Recall that Algorithm FMIS (as discussed in Section 4.6) takes time $n^{O(l)}$ for computing an independent set of size at least $(l/(l+1))|OPT|$ for a set of n unit disks. Thus, the time required for executing Step 1 is $N^{O(l^2)}$.

Next consider each iteration of Step 2. The number of vertices (explicit vertices and nonterminals) in $PE(G_i^l)$ is $O(N^{l+1})$. By arguments similar to those presented for analyzing Step 1 it follows that the running time for each iteration of Step 2 is $N^{O(l^2)}$. Thus, the running time of Step 2 is $N \cdot N^{O(l^2)} = N^{O(l^2)}$. Similarly, the running time of Step 3 is also $N^{O(l^2)}$. ∎

The heuristics for the other problems work in a similar fashion. In the case of minimization problems, instead of deleting nodes at levels $j = i$ mod$(l+1)$, we consider them in both sides of the partition. The reader can easily verify that this can be done by slightly modifying the definition of *partial expansion*. Hence by a straightforward combination of our ideas in Section 4.7 together with the ideas mentioned in this section we can prove the following theorems.

THEOREM 6.11. *For all fixed $k \geq 0$ and $l \geq 1$, given a k-near-consistent BOW-specification Γ of a set of unit disks, there are $N^{O(l^2)}$ time approximation algorithms with performance guarantee $((l+1)/l)^2$ for the problems maximum independent set, minimum vertex cover, and minimum edge dominating set, where N is the size of Γ.*

For λ-precision unit disk graphs, we can obtain approximation schemes for many more problems and also obtain a better time performance trade-offs. This is summarized in the following theorem.

THEOREM 6.12. *For all fixed $k \geq 0$ and $\lambda > 0$, given a k-near-consistent BOW-specification of a set of λ-precision unit disks, there are $N^{O(l)}$ time approximation algorithms with performance guarantee $((l+1)/l)^2$ for the following problems: maximum independent set, minimum vertex cover, maximum H-matching, minimum edge dominating set, and maximum cut.*

Proof Sketch. We again consider MIS for purposes of illustration. Note that when λ-precision unit disks are specified using standard specifications, we have an $O(2^l n)$ time approximation algorithm with performance guarantee $(l+1)/l$ for instances of size n. This fact in conjunction with the arguments used to prove Theorem 6.11 yields the required result. ∎

The above ideas can be easily extended along the lines of the results in [MH + 94] to solve the query, output, and the construction problems (discussed in Section 6.2) associated with each optimization problem considered here. As the reader can note, an approximation scheme for the minimum dominating set problem is not claimed in Theorems 6.11 and 6.12. It is not clear at this time how to obtain such a scheme.

7. EXTENSIONS

We discuss two extensions of the preceding results. First, we discuss briefly the ideas behind parallelizing Algorithm HMIS. Similar ideas hold for obtaining parallel algorithms for other problems for near-consistent BOW-specifications.

For any fixed $l \geq 1$, the size of any partially expanded graph is $O(N^l)$. Such an expansion can easily be obtained in NC. Furthermore, we know from the results in Section 4 that there is an NC-approximation scheme for the MIS problem for unit disk graphs. Therefore, Steps 2(a) and 2(b) of Algorithm HMIS can be implemented in NC. Next consider Step 2(c). In the sequential case, we could evaluate Step 2(c) for nonterminals starting from G_1. Implementing Step 2(c) in NC is a bit more subtle but can be done using classical techniques for parallel prefix computation. To do this, we first abstract the basic problem we need to solve.

Recall that there is a hierarchy tree representing the sequence of calls that are made by the nonterminals. Note that the tree can be exponentially larger than the size of the specification. Hence a direct application of parallel prefix algorithm on this tree will not yield an NC-algorithm. However, we can overcome this difficulty by observing that there are efficient NC-algorithms for solving higher order recurrences. We now discuss our ideas in some detail.

Consider the 1-near-consistent BOW-specification Γ. The specification can be seen to be a restricted form of context-free graph grammar. An additional restriction imposed on the graph grammar to obtain 1-near-consistent BOW-specified graphs is that for each nonterminal there is only one cell that can be substituted. Thus there are no *alternatives* for substitution. Also, the index of the substituted cell has to be *smaller* than the index of the cell in which the nonterminal occurs. This *acyclicity* condition implies that the 1-near-consistent BOW-specification defines a unique graph. Consider the hierarchy tree corresponding to the specification Γ after computing the partial expansion of each of the nonterminals. Using our equation in Step 2(c) of the algorithm it follows that the maximum independent set in the graph $E(G_i)$ can be calculated by the following higher order recurrence.

$$HIS(G_i) = \begin{cases} b_i, & 1 \le i \le l, \\ \left(HIS(G_{i-1}) \times a_{i-1}\right) & \\ \quad + \cdots + \left(HIS(G_1) \times a_1\right) + b_i, & l + 1 \le i \le n - 1. \end{cases}$$

In the recurrence equations above, a_k denotes the number of copies of nonterminals G_k called in G_i, and b_i denotes the size of the near optimal independent set for the set of explicit vertices $Ex(PE(G_i^l))$. Such a system of higher order recurrences has an NC-algorithm as discussed in [Re93, Sect. 1.4.2, p. 50]. This completes the discussion of how Step 2 can be parallelized.

Next consider each iteration of Step 3. It is easy to see that Step 3 can be executed in NC. The total number of iterations is $(l + 1)$. Combining the above ideas, it is easy to obtain an NC-approximation scheme for the maximum independent set problem for 1-near-consistent BOW-specified unit disk graphs.

By similar arguments we get that, for all the problems Π listed in Table II (see Section 8), there exist NC-approximation schemes when instances are specified using k-near consistent BOW-specifications, for any fixed $k \ge 1$.

We end this section by presenting another extension of the definition of k-near-consistent specifications for which the problems are still approximable.

DEFINITION 7.1. A BOW specification is 1-extended-near-consistent iff for each symbol G_i the following conditions hold:

1. The MBR of G_i contains the MBR of all the symbols called in G_i.

2. Let G_i call symbols $G_{i_1}, \dots, G_{i_k}$. The MBR of the symbol G_{i_p}, $1 \leq p \leq k$, called in G_i does not intersect the MBR of a symbol G_m such that G_m is called in some G_{i_q}, $q \neq p$.

3. For each explicit disk u defined in G_i, u does not intersect the MBR of any symbol G_k such that G_k occurs in $HT(G_i)$ and level number of G_k in $HT(G_i)$ is ≥ 2.

The difference between Definitions 6.2 and 7.1 is the statement of Condition 2. In Definition 7.1 we relax the condition that the MBR of the called symbols are nonintersecting and allow for intersections of MBRs of two symbols which are *siblings in the hierarchy tree*. It is not difficult to verify the following.

1. We can check in polynomial time if a given BOW-specification is 1-extended-near-consistent.

2. Lemma 6.4 holds.

3. The size of each graph obtained as a result of partial expansion is polynomial in N.

With these properties, it is easy to see that our approximation schemes can be extended so as apply to k-extended-near-consistent specifications, for any fixed $k \geq 1$.

8. CONCLUSIONS

We have presented efficient approximation schemes for a large class of geometric intersection graphs when the graphs are represented using a geometric layout. All our results are based on the shifting technique. This technique was further generalized to obtain approximation schemes for PSPACE-hard problems for hierarchically specified geometric intersection graphs. Our results are summarized in Tables I and II. As in [GJ79], we use the convention that the performance guarantees for minimization as well as maximization problems are always at least 1. We believe that the approximation schemes obtained in this paper demonstrate the following additional insights:

1. The crucial property of planar graphs used to obtain efficient approximation schemes for problems which can be solved approximately by a divide-and-conquer type of algorithm is that any planar graph can be decomposed into a disjoint collection of subgraphs for which the given problems considered can be solved exactly and efficiently. Any graph class with this property is amenable to a similar approximation scheme.

TABLE I
Performance Guarantees for Geometric Intersection Graph Problems[a]

Nonhierarchical				
Problem	λ-precision	Civilized	Unit disk	Isothetic squares
MIN VC	$\left(\frac{k+1}{k}\right)$	$\left(\frac{k+1}{k}\right)$	$\left(\frac{k+1}{k}\right)^2$	$\left(\frac{k+1}{k}\right)^2$
MAX IS	$\left(\frac{k+1}{k}\right)$	$\left(\frac{k+1}{k}\right)$	$\left(\frac{k+1}{k}\right)^2$	$\left(\frac{k+1}{k}\right)^2$
MIN Dom Set	$\left(\frac{k+1}{k}\right)$	$\left(\frac{k+1}{k}\right)$	$\left(\frac{k+1}{k}\right)^2$	$\left(\frac{k+1}{k}\right)^2$
MAX partition into triangles	$\left(\frac{k+1}{k}\right)$	$\left(\frac{k+1}{k}\right)$	3 [CK94]	3 [CK94]
MAX CUT	$\left(\frac{k+1}{k}\right)$	$\left(\frac{k+1}{k}\right)$	1.137 [GW94]	1.137 [GW94]
MIN edge dom set	$\left(\frac{k+1}{k}\right)$	$\left(\frac{k+1}{k}\right)$	2 [CK94]	2 [CK94]
MAX H-matching	$\left(\frac{k+1}{k}\right)$	$\left(\frac{k+1}{k}\right)$	Open	Open

[a] The parameter k can be any fixed integer ≥ 1.

2. The results also provide a better understanding of the close relationship between planar graphs and intersection graphs of unit disks. Clark *et al*. [CCJ90] pointed out that from a complexity theoretic point of view, unit disk graphs appeared to be closer to planar graphs than to grid graphs. We have provided additional evidence in support of the above statement by presenting approximation schemes for many problems on unit disk graphs, *all* of which are known to have approximation schemes when restricted to planar graphs.

The results and techniques presented here have been recently applied to other problems arising in pattern matching and map labeling [AD + 97, DM + 97]. The results have also been extended to apply to a large set of predicates and also to a larger class of graphs [KM96, MHS97].

The results presented in this paper raise several open questions. First, is it possible to devise polynomial time approximation schemes with similar time−performance trade-offs for problems restricted to circle intersection graphs? *Circle intersection graphs* are intersection graphs of circles (of arbitrary radii) in which we include an edge between a pair of vertices only when the corresponding circles intersect. A subset of circle intersection

TABLE II
Performance Guarantees for k-Near-Consistent Specifications[a]

k-near-consistent BOW-specifications			
Problem	λ-precision	Unit disk	Isothetic squares
MIN VC	$\left(\frac{l+1}{l}\right)^2$	$\left(\frac{l+1}{l}\right)^3$	$\left(\frac{l+1}{l}\right)^2$
MAX IS	$\left(\frac{l+1}{l}\right)^2$	$\left(\frac{l+1}{l}\right)^3$	$\left(\frac{l+1}{l}\right)^3$
MAX partition into triangles	$\left(\frac{l+1}{l}\right)^2$	3	3
MAX CUT	$\left(\frac{l+1}{l}\right)^2$	2	2
MIN edge dom set	$\left(\frac{l+1}{l}\right)^2$	$2\left(\frac{l+1}{l}\right)^2$	$2\left(\frac{l+1}{l}\right)^2$
MAX H-matching	$\left(\frac{l+1}{l}\right)^2$	Open	Open

[a]The parameter l can be any fixed integer ≥ 1.

graphs is the class of *coin graphs* (also referred to as *sphere packing graphs*) defined as follows.

DEFINITION 8.1. A *sphere packing* in d dimensions is a set of spheres having disjoint interiors. A *sphere packing graph* is a graph with vertices which are in 1−1 correspondence with the spheres and an edge between two vertices if the corresponding spheres touch.

In addition to their applications in communication networks, circle intersection graphs have been a subject of interest to researchers lately due to their relationship with planar graphs. In particular, Andreev and Thurston [Th88] showed that

THEOREM 8.1. *Every triangulated planar graph is isomorphic to a two-dimensional sphere packing graph.*

It can be seen from the preceding definitions that circle intersection graphs contain both planar graphs (coin graphs) and unit disk graphs as special cases. Hence, it is of interest to investigate the complexity of finding good approximation algorithms for basic graph problems restricted to circle intersection graphs and intersection graphs of other geometric objects. As a step in this direction, in [MB + 95] we gave simple approximation algorithms with constant performance guarantees for several problems on circle intersection graphs. In particular, we showed that minimum dominating set and maximum independent set can be approximated to within a factor of 5, minimum vertex coloring can be approximated to within a factor of 6, and minimum vertex cover can be approximated to within a factor of 5/3 of the optimal. These approximation algorithms do not require geometric input.

Existence of approximation schemes for circle intersection graphs would generalize the results presented in this paper as well as those in [Ba94]. If this is not possible, one would like to devise approximation schemes for a nontrivial subclass of circle intersection (and in general geometric intersection) graphs that contains both coin graphs (planar graphs) and unit disk graphs.

ACKNOWLEDGMENTS

We thank Professor David Peleg (Weizmann Institute) for carefully reading the manuscript and pointing out an error in our approximation scheme for the minimum dominating set problem. We thank Professor R. Ravi (Carnegie Mellon University) and Dr. Ravi Sundaram (Delta Global Trading, Inc.) for constructive suggestions. We also thank Professor S. H. Teng (University of Minnesota) and Dr. S. Ramanathan (BBN) for making available copies of their theses, and Professor Tao Jiang (McMaster University) for making available a copy of his paper. The second author thanks Professors Thomas Lengauer (GMB, Bonn) and Egon Wanke (University of Dusseldorf) for fruitful discussions on succinct specifications.

REFERENCES

[AD + 97] S. R. Arikati, A. Dessmark, A. Lingas, and M. V. Marathe, Approximation algorithms for maximum two-dimensional pattern matching, *in* "Proc. 7th Annual Symposium on Combinatorial Pattern Matching (CPM), Laguna Beach, CA, June 1996," Lecture Notes in Computer Science, Vol. 1075, pp. 348–360, Springer-Verlag, Berlin/New York. (A complete version of the paper appears as Technical Report LA-UR-96-1375, Los Alamos National Laboratory, 1996.)

[ALS91] S. Arnborg, J. Lagergren, and D. Seese, Easy problems for tree-decomposable graphs, *J. Algorithms* **12** (1991), 308–340.

[AP89] S. Arnborg and A. Proskurowski, Linear-time algorithms for NP-hard problems on graphs embedded in k-trees, *Discrete Appl. Math.* **23** (1989), 11–24.

[Ba94] B. S. Baker, Approximation algorithms for NP-complete problems on planar graphs, *J. Assoc. Comput. Mach.* **41**(1) (1994), 153–180.

[BK93] H. Breu and D. Kirkpatrick, "Unit Disk Graph Recognition is NP-hard," Technical Report 93-27, Department of Computer Science, University of British Columbia, August, 1993.

[Bo88] H. L. Bodlaender, Dynamic programming on graphs of bounded treewidth, *in* Proc. 15th International Colloquium on Automata Languages and Programming (ICALP)," Lecture Notes in Computer Science, Vol. 317, pp. 105–118, Springer, Verlag, Berlin/New York, 1988.

[BOW83] J. L. Bentley, T. Ottmann, and P. Widmayer, The complexity of manipulating hierarchically defined sets of rectangles, *in* Advances in Computing Research (F. P. Preparata, Ed.), Vol. 1, pp. 127–158, JAI Press, Greenwich, CT, 1983.

[CCJ90] B. N. Clark, C. J. Colbourn, and D. S. Johnson, Unit disk graphs, *Discrete Math.* **86** (1990), 165–177.

[CF + 93] A. Condon, J. Feigenbaum, C. Lund, and P. Shor, Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions, *Chicago J. Theoret. Comput. Sci.* **1995**(4) (http://www.cs.uchicago.edu/publications/cjtcs/articles/1995/4/contents.html).

[CF + 94] A. Condon, J. Feigenbaum, C. Lund, and P. Shor, Random debators and the hardness of approximating stochastic functions, *SIAM J. Comput.* **26**(2) (1997), 369–400.

[CK94] P. Crescenzi and V. Kann, A compendium of NP-optimization problems, preliminary version, March 1997.

[CLR91] T. Cormen, C. E. Leiserson, and R. L. Rivest, ‘‘Introduction to Algorithms,’’ MIT Press, Cambridge, MA, 1991.

[DM + 97] S. Doddi, M. V. Marathe, A. Mirzaian, B. Moret, and B. Zhu, Map labeling problems, *in* ‘‘Proc. 8th ACM-SIAM Symposium on Discrete Algorithms (SODA), 1997, pp. 148–157.

[DS84] P. G. Doyle and J. L. Snell, ‘‘Random Walks and Electric Networks,’’ The Carus Mathematical Monographs, Math. Assoc. of America, Washington, DC, 1984.

[DST96] J. Diaz, M. J. Serna, and J. Toran, Parallel approximation schemes for problems on planar graphs, *Acta Inform.* **33**(4) (1996), 387–408.

[EMT93] D. Eppstein, G. L. Miller, and S. H. Teng, A deterministic linear time algorithm for geometric separators and its application, *in* ‘‘Proc. 9th ACM Symposium on Computational Geometry, 1993,’’ pp. 99–108.

[Ep95] D. Eppstein, Subgraph isomorphism in planar graphs and related problems, *in* ‘‘Proc. 6th ACM-SIAM Symposium on Discrete Algorithms (SODA), 1995,’’ pp. 632–640.

[FG88] T. Feder and D. Greene, Optimal algorithms for approximate clustering, *in* ‘‘30th ACM Symposium on Theory of Computing (STOC), 1988,’’ pp. 434–444.

[FPT81] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, Optimal packing and covering in the plane are NP-complete, *Inform. Process. Lett.* **12**(3) (1981), 133–137.

[GJ79] M. R. Garey and D. S. Johnson, ‘‘Computers and Intractability: A Guide to the Theory of NP-completeness,’’ Freeman, San Francisco, CA, 1979.

[GSW94] A. Gräf, M. Stumpf, and G. Weisenfels, On coloring unit disk graphs, unpublished manuscript, July 1994.

[GW94] M. X. Goemans and D. P. Williamson, .878 Approximation Algorithms for MAX CUT and MAX 2SAT, *in* ‘‘Proc. 26th Annual ACM Symposium on Theory of Computing (STOC), May 1994,’’ pp. 422–431.

[Ha80] W. K. Hale, Frequency assignment: Theory and applications, *Proc. IEEE* **68** (1980), 1497–1514.

[HM85] D. S. Hochbaum and W. Maass, Approximation schemes for covering and packing problems in image processing and VLSI, *J. Assoc. Comput. Mach.* **32**(1) (1985), 130–136.

[HM87] D. S. Hochbaum and W. Maass, Fast approximation algorithms for a nonconvex covering problem, *J. Algorithms* **8** (1987), 305–323.

[HM + 93] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns, Designing approximation schemes using L-reductions, *in* ‘‘Proceedings of 14th Foundations of Software Technology and Theoretical Computer Science (FST & TCS),’’ Lecture Notes in Computer Science, Vol. 761, pp. 342–353, Springer-Verlag, Berlin/New York, 1994. (A complete version of the paper, titled "Parallel Approximation Schemes for Planar and Near-Planar Satisfiability and Graph Problems," is available as Technical Report LA-UR-96-2723, Los Alamos National Laboratory, 1996.)

[JW94] J. Jiang and L. Wang, An approximation scheme for some Steiner tree problems in the plane, *in* "International Symposium on Algorithms and Computation (ISAAC), 1994."

[Ka84] K. Kammerlander, C 900—an advanced mobile radio telephone system with optimum frequency utilization, *IEEE Trans. Selected Areas in Communication* **2** (1984), 589–597.

[KM96] S. Khanna and R. Motwani, Towards a syntactic characterization of PTAS, *in* "Proc. 28th Annual ACM Symposium on Theory of Computing, (STOC), 1996," pp. 329–337.

[KS93] P. N. Klein and S. Sairam, An efficient parallel algorithm for shortest paths in planar graphs, *in* "Proc. 34th IEEE Symposium on Foundations of Computer Science (FOCS), 1993," pp. 259–270.

[LT79] R. Lipton and R. E. Tarjan, A separator theorem for planar graphs, *SIAM J. Appl. Math.* **32**(2), (1979), 177–189.

[LW87a] T. Lengauer and E. Wanke, Efficient solutions for connectivity problems for hierarchically defined graphs, *SIAM J. Comput.* **17**(6) (1988), 1063–1080.

[LW92] T. Lengauer and K. W. Wagner, The correlation between the complexities of nonhierarchical and hierarchical versions of graph problems, *J. Comput. System Sci.* **44** (1992), 63–93.

[MB + 95] 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), 59–68.

[MC80] C. A. Mead and L. Conway, "Introduction to VLSI Systems," Addison-Wesley, Reading, MA, 1980.

[MHS97] M. V. Marathe, H. B. Hunt III, and R. E. Stearns, "Level Treewidth Property: Exact Algorithms and Approximation Schemes," Technical Report LA-UR-97-479, Los Alamos National Laboratory, Jan. 1997.

[MH + 94] M. V. Marathe, H. B. Hunt III, R. E. Stearns, and V. Radhakrishnan, Approximation schemes for PSPACE-complete problems for succinct graphs, *in* "Proceedings of 26th Annual ACM Symposium on the Theory of Computing (STOC), May 1994," pp. 468–477. (A complete version of the paper, entitles "Approximation Algorithms for PSPACE-Hard Hierarchically and Periodically Specified Problems," will appear in *SIAM J. Comput.*, 1998.)

[MH + 94b] M. V. Marathe, H. B. Hunt III, R. E. Stearns, and V. Radhakrishnan, "Complexity of Hierarchically and 1-Dimensional Periodically Specified Problems," Technical Report LA-UR-95-3348, Los Alamos National Laboratory, August 1995. (To appear in "Proc. DIMACS Workshop on Satisfiability Problem: Theory and Applications," Amer. Math. Soc., Providence, 1997.)

[MHR94] M. V. Marathe, H. B. Hunt III, and S. S. Ravi, The complexity of approximating PSPACE-complete problems for hierarchical specifications, *Nordic J. Comput.* **1** (1994), 275–316.

[MR + 97] M. V. Marathe, V. Radhakrishnan, H. B. Hunt III, and S. S. Ravi, Hierarchically specified unit disk graphs, *Theoret. Comput. Sci.* **174**(1–2) (1997), 23–65.

[MS84] N. Meggido and K. Supowit, On the complexity of some common geometric location problems, *SIAM J. Comput.* **13**(1) (1984), 182–196.

[MT + 97] G. L. Miller, S. H. Teng, W. Thurston, and S. A. Vavasis, Separators for sphere packings and nearest neighbor graphs, *J. Assoc. Comput. Mach.* **44**(1) (1997), 1–29.

[Pe91] R. Peeters, ‘‘On Coloring *j*-unit Sphere Graphs,’’ FEW 512, Department of Economics, Tilburg University NL, 1991.

[Re93] J. Reif, Ed., ‘‘Synthesis of Parallel Algorithms,’’ Morgan Kaufmann, San Mateo, CA, 1993.

[Te91] S. H. Teng, ‘‘Points, Spheres, and Separators, A Unified Geometric Approach to Graph Separators,’’ Ph.D. thesis, School of Computer Science, Carnegie Mellon University, CMU-CS-91-184, August 1991.

[Th88] W. P. Thurston, ‘‘The Geometry and Topology of 3-manifolds,’’ Princeton University Notes, 1988.

[Va91] S. A. Vavasis, Automatic domain partitioning in three dimensions, *SIAM J. Sci. Statist. Comput.*, 1991.

[YWS84] Y. Yeh, J. Wilson, and S. C. Schwartz, Outage probability in mobile telephony with directive antennas and macrodiversity, *IEEE Trans. Selected Areas in Communication* **2** (1984), 507–511.

[WK88] D. W. Wong and Y. S. Kuo, A study of two geometric location problems, *Inform. Process. Lett.* **28**(6) (1988), 281–286.