Local Algorithms for Dominating and Connected Dominating Sets of Unit Disk Graphs

J. Czyzowicz[‡‡**] S. Dobrev[†**] T. Fevens[§ **] H. González-Aguilar[††]
E. Kranakis[‡**††] J. Opatrny[§ **] J. Urrutia[‖]

January 19, 2007

Abstract

Many protocols in distributed computing make use of dominating and connected dominating sets, for example for broadcasting and the computation of routing. Ad hoc networks impose an additional requirement that algorithms for the construction of such sets should be *local* in the sense that each node of the network should make decisions based only on the information obtained from nodes located a constant (independent of the size of the network) number of steps away from it. The focus of the present paper is on providing local, constant approximation, deterministic algorithms for the construction of dominating and connected dominating sets of a Unit Disk Graph (UDG) with location aware nodes (i.e., nodes that know their coordinates in the plane). The size of the constructed set, in the case of the dominating set, is shown to be 5 times the optimal, while for the connected dominating set $7.453 + \epsilon$ the optimal, for any arbitrarily small $\epsilon > 0$. These are the first *local* algorithms in the scientific literature whose time complexities and approximation bounds are completely independent on the size of the network.

Key Words and Phrases: Approximation factor, Connected dominating set, Dominating set, Local algorithm, Location aware, Unit disk graph.

[‡‡]Département d'informatique, Université du Québec en Outaouais, Gatineau, Québec J8X 3X7, Canada

[†]School of Information Technology and Engineering (SITE), University of Ottawa, 800 King Eduard, Ottawa, Ontario, Canada, K1N 6N5. (On leave from the Slovak Academy of Sciences.)

[††]Centro de Investigacion en Matematicas, Guanajuato, Gto., C.P. 36000, Mexico

[§]Department of Computer Science, Concordia University, 1455 de Maisonneuve Blvd West, Montréal, Qubec, Canada, H3G 1M8.

[‡]School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6.

[‖]Instituto de Matemáticas, Universidad Nacional Autónoma de México, Área de la investigación cientifica, Circuito Exterior, Ciudad Universitaria, Coyoacán 04510, México, D.F. México.

[**]Research supported in part by NSERC (Natural Science and Engineering Research Council of Canada).

[††]Research supported in part by MITACS (Mathematics of Information Technology and Complex Systems).

1 Introduction

Many of the existing networks have become large and complex. The number of connections (links) between nodes often remains relatively small, each node being capable to communicate, on average, with a limited number of neighbors. Distributed algorithms implemented over such networks must often achieve some *global* computational tasks, like computing good approximations of dominating and independent sets, vertex and edge colorings, spanners, etc., despite the fact that each node is confined to *local* communication. Consequently, in the last twenty years, local algorithms have been investigated by several researchers in distributed computing. For example, in a k-local algorithm, for a given parameter k, a node is allowed to communicate at most k times with its neighbors. Work on this model includes Luby's randomized independent set algorithm Luby [1985], sparse partitions introduced by Awerbuch and Peleg [1990] and particularly the seminal work of Linial [1992].

A distributed algorithm is called *local* if each node of the network makes decisions based on the information obtained uniquely from the nodes located no more than a constant (independent of the size of the network) number of hops from it. Thus, during the algorithm, no node is ever aware of the existence of the parts of the network further away than this constant number of hops. According to the model of Linial [1992], local algorithm implies constant time complexity. There are several reasons why such local algorithms are practical for wireless, ad hoc networks. The most important include the following.

1. A solution is consistent regardless of the order in which the nodes or edges are considered in the calculations.
2. Changes in the network outside of a fixed-size neighborhood do not influence the computation. Moreover, adapting to a change in the network requires solely a local recalculation of the solution.
3. It is possible to calculate only a part of the required subnetwork that is really needed without necessarily having to calculate a complete solution (this can be important in cases of disaster recovery).
4. Messages do not propagate indefinitely throughout the network and the algorithm terminates in a constant number of steps.

Wireless ad hoc and sensor networks are most often modeled by *Unit Disk Graphs* (abbreviated by UDGs). Nodes of a UDG are located on a plane and two nodes are considered adjacent when their distance is at most equal to some given constant. Hence it is assumed that the wireless nodes have equal transmission range and two nodes can communicate if they are inside each other's transmission range. Most NP-hard graph theory problems remain NP-hard when restricted to the class of UDGs. However, it turns out that for the class of UDGs it is possible to design algorithms offering better approximative solutions.

Many graph-theoretic problems do not admit local algorithms solving them, even if restricted to the class of UDGs. Conversely, it turns out, that several of these problems become solvable in the local setting when the network is *location aware*, i.e. when the graph is embedded in the plane and each node knows its *geographic position* (e.g. Cartesian coordinates). Notwithstanding the cost associated with the possession of such *geographic information*, it also happens that algorithms for location aware networks are sometimes easier to design and they may lead to better time complexities and/or approximation bounds. The recent survey of open problems by Aspnes et al. [2006] lists the problem of local computation of dominating sets for UDGs as one of the important open problems in distributed computing.

In this paper we consider the problems of minimum dominating set and minimum connected dominating set for a location aware network, represented by a UDG graph. We design local algorithms (i.e. working in constant time) providing constant approximation solutions to both problems. To the best of our knowledge this is the first solution when both of these parameters, i.e. time complexity and approximation bounds are completely independent on the size of the network.

1.1 Related work

It is well known that the dominating set and connected dominating set problems are NP-hard, even when restricted to the class of UDGs (cf. Clark et al. [1990]). The importance of these problems have motivated researchers to investigate approximation schemes. For general graphs there exists an $O(\log n)$ approximation algorithm for the minimum dominating set problem, cf. Johnson [1974], and it is known that no polynomial-time approximation of $o(\log n)$ exists unless every problem in NP can be solved deterministically in $O(n^{poly \log n})$ time, cf. Lund and Yannakakis [1994]. For general graphs it is also known that, unless $P = NP$, there is an $\epsilon > 0$ such that the minimum independent dominating set cannot be approximated within a factor of $O(n^{\epsilon})$, cf. Irving [1991].

The situation is quite different in the case of UDGs. Despite the fact that the dominating set and connected dominating set problems for the class of unit disk graphs remain NP-hard, constant approximation is possible (e.g. Marathe et al. [1994], Alzoubi et al. [2002]), even polynomial-time approximation schemes (PTAS) are known for this case. The first such solution has been proposed by Hunt III et al. [1998], where the geometric representation of the UDG was supposed to be part of the input. Additionally, Nieberg and Hurink [2005] have given a PTAS for the minimum dominating set problem of a UDG for which the geometric representation was not given.

The approximation bounds, which appear to be better when restricted to the class of UDGs apply mainly to the centralized setting. In the distributed scenario, however, especially if the geometric information of the input UDG is not given, the best approximation of the minimum dominating set problem is often not better than one obtained for the case of general graphs. The first algorithm achieving a nontrivial approximation ratio $o(\Delta)$, for Δ being the maximum node degree, in a nontrivial time $o(diam(G))$ was developed in Jia et al. [2002]. Kuhn and Wattenhofer [2005] proposed a distributed approximation based on LP relaxation techniques. Kuhn et al. [2005a] designed a PTAS for the minimum dominating set for the class of graphs of *polynomially bounded growth*. In such graphs in the k-neighborhood of any node the size of an independent set is bound by $f(k)$ (where $f(k)$ is a polynomial function) and they include the class of UDGs.

Since the time of the pioneering work of Linial [1992] on locality in distributed computing many papers on local algorithms have been published. However, despite the fact that several lower bounds and impossibility results in distributed computing are now known (e.g. Fich and Ruppert [2003]), most of them apply to the computational models which do not involve locality. In fact, for a long time, the only nontrivial lower bound in local distributed computing, known to the authors of this paper has been $\Omega(\log^* n)$ time for 3-coloring of the ring by Linial. On the other hand, it was shown in Naor and Stockmeyer [1995] that there are nontrivial *Locally Checkable Labeling* (LCL) problems having local, i.e. constant-time solutions. Peleg [2000] discussed several problems in distributed computing in the context of a *locality-sensitive approach*. For the class of general graphs Kuhn et al. [2004a] have given approximation lower bounds for covering problems as a function of the size of the neighborhood through which each message may be propagated. In the case of UDGs (or a more general class of *bounded growth graphs*), besides Kuhn et al. [2005a] mentioned above, Kuhn et al. [2005b] have given local approximation algorithms for the class of covering and packing linear programs. Nevertheless, the recent paper of Kuhn et al. [2006], providing the trade-offs between the amount of local information used and the quality of the global solution for general graphs, offer the best solution for the dominating set problem for the class of UDGs.

The local algorithms mentioned above are such that, either the approximation bounds proposed, or the worst case time bounds obtained depend on the size n of the network. However, these algorithms do not use the underlying geographic information. In this paper we prove that these bounds are not always valid when the geographic information is allowed to be part of the input, i.e. when each node is aware of its cartesian coordinates in the plane. It has been known that the algorithms not using the geographic information are more difficult to design. Since finding a geographic representation of a given UDG graph (cf. Breu and Kirkpatric [1998]) or even its approximation (cf. Kuhn et al. [2004b]) is known to be NP-hard it seems that the geographic form

of the input data is a powerful information indeed and using it may result in better algorithmic bounds. For example, for a different problem of broadcasting in the *geometric radio networks* Dessmark and Pelc [2007] have shown an $O(n)$ time algorithm using the geographic information and $\Omega(n \log n)$ time lower bound when the geographic information was not available. With the recent development of GPS systems the cost of geographic awareness is well justified by better algorithmic bounds. In this paper we present two local, constant time distributed algorithms producing constant approximations of minimum dominating sets and minimum connected dominating sets, respectively.

1.2 Preliminaries and results of the paper

Consider a graph $G(V, E)$ with vertex set V and edge set E. A subset S of V is called *dominating set* if every vertex of G is either in S or adjacent to a vertex in S. A dominating set S is called a *connected dominating set* if the subgraph of G induced by S is connected. S is an *independent set* if there is no edge of G between any two elements of S.

The focus of the present paper is to provide a local, constant approximation algorithm for the construction of a dominating set as well as connected dominating set of a unit disk graph. We assume that a wireless network consists of nodes that have the same *circular* transmission range of size 1. Thus, it can be represented by a UDG with an edge connecting two nodes when they are at most a unit distance from each other. We assume that the network is *location aware*, i.e. each node of the graph knows its geometric position in the plane. We suppose that at each time unit a vertex may send a message to each of its neighbors.

In this paper we give two deterministic, local, constant approximation algorithms. In Section 2 we give an algorithm for the construction of a dominating set of a unit disk graph. The algorithm is shown to have a competitive ratio 5 (Theorem 1). Section 3 gives an algorithm for the construction of a connected dominating set of a unit disk graph. This algorithm has a competitive ratio $7.453 + \epsilon$, for any $\epsilon > 0$ (Theorem 2). To the best of our knowledge these are the first such algorithms whose time complexities and approximation parameters are independent of the size of the network.

2 Local Algorithm for Dominating Set of a UDG

As previously noted, given a UDG a local algorithm decides to include a node into a dominating set using a fixed size neighborhood independently of decisions possibly taken at other nodes. Thus the algorithm could actually construct a very large dominating set due to symmetries that could be present in the graph. We therefore need to make sure that potential symmetries in the graph are broken in some way. Using the fact that the nodes are aware of their coordinates in the plane we associate with each node a class number that depends on the position of the node within a regular tiling of the plane. As depicted in Figure 1, for the tiling used by our algorithm (see Algorithm 1) each tile consists of 12 hexagons of unit diameter and each hexagon of the tile represents a single class. In a hexagon we assume that its right-hand side boundary, starting from the top apex of the hexagon up to the bottom apex of the hexagon belongs to the hexagon; thus, only the top apex and the two right upper apexes are considered to belong to the hexagon (see Figure 2).

We assume that the tiling starts by placing one tile with the center of the hexagon of class number 1 in coordinates $(0, 0)$, while other tiles are placed so that the hexagon of Class 3 is made adjacent to hexagons of classes 7 and 10 or the hexagon of Class 11 is made adjacent to hexagons of Classes 8 and 4, etc. Figure 3 depicts the tiling of the plane that is used in our algorithm. The following Lemma 1 provides an important separation property of the hexagons of the tiling that is useful in the sequel.

Lemma 1 *In the tiling of the plane given above, any two points of the plane that are of the same class, but belong to two different hexagons, are at Euclidean distance greater than 2. Moreover,*

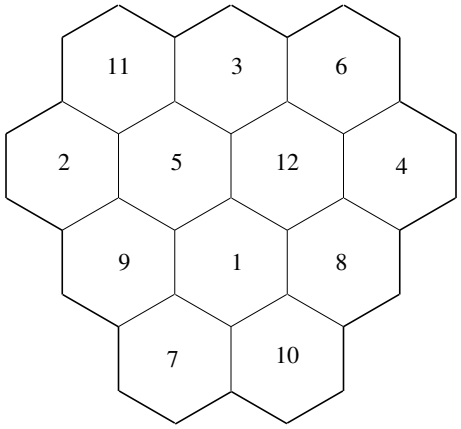

Figure 1: Image depicts a tile divided into 12 hexagons of diameter 1 and the class numbering associated with the hexagons.

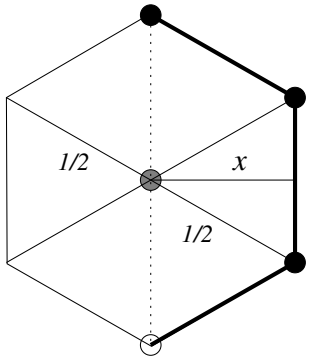

Figure 2: Image depicts the dimensions of the hexagon, while the "bold" lines illustrate the boundary that belongs to this hexagon.

given the coordinates of a point P in the plane, one can determine the class number of P using a calculation of constant cost.

Proof. Without loss of generality consider the hexagon of Class 1 in the central tile of Figure 3 and the hexagons of Class 1 in the six adjacent tiles. Figure 2 depicts the lengths of a unit hexagon which are relevant to the calculations. In particular, the perpendicular distance x from the center of the hexagon to the opposite side is clearly equal to $\sqrt{3}/4$. It is easy to calculate from Figure 4 that if the coordinates of the hexagon X at the center are (a, b) then the coordinates of the neighboring hexagons of the same class must be $A = (a, b + 3), B = (a + 3\sqrt{3}/2, b + 3/2), C = (a + 3\sqrt{3}/2, b - 3/2), D = (a, b - 3), E = (a - 3\sqrt{3}/2, b - 3/2), F = (a - 3\sqrt{3}/2, b + 3/2)$. It is also easy to verify that all points within these six hexagons are at distance two or more from the hexagon in the central tile. Moreover, it is simple to determine the lattice points whose centers are the hexagons of class 1 as coordinate translations of the class 1 hexagons. Since only the top apex and the right upper apex belong to a tile, there is no pair at distance exactly two.

The tiling of the plane depicted in Figure 3 is achieved by appropriate geometric translation of the basic tile in Figure 1. Due to the symmetry of the tiling used, the distance requirements just proved for class 1 hexagons apply to any other class number. Moreover, given the coordinates of a

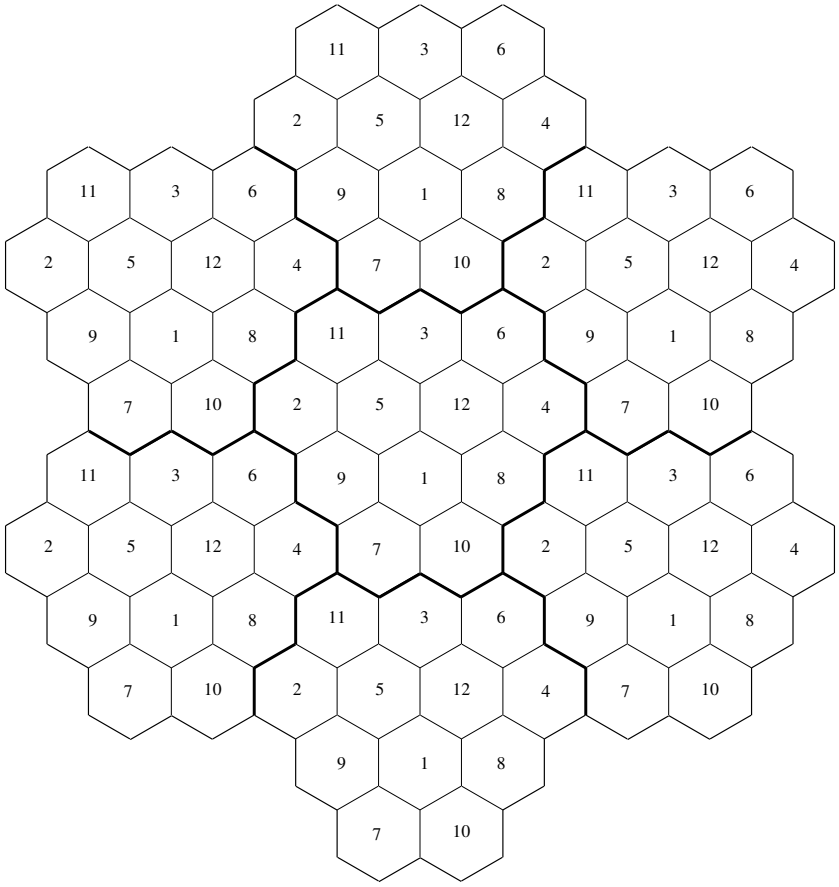

Figure 3: Tiling of the plane with tiles consisting of 12 hexagons each.

node P, one can calculate from its coordinates, using only a local calculation of constant cost, the unique class number the node belongs to. This completes the proof of Lemma 1. ■

As a consequence of Lemma 1, if all nodes of a network are aware of the tiling being used, then any node can calculate from its own coordinates and using only a local calculation of constant cost, the unique class number the node belongs to. In the following discussions, this knowledge of the tiling as well as class numbering being used is assumed to be available to each node. Also note that this is a constant size information.

2.1 Construction of the dominating set

The main idea of our algorithm for computing locally a dominating set is as follows. Nodes determine their class number and acquire the class numbers of all their neighbors. In each hexagon, dominators are determined on the basis of unassigned neighbors of minimum class number closest to the center of the hexagon under consideration. More precisely, to calculate a dominating set of a unit disk graph G, each node of G executes Algorithm 1.

Let $\mathcal{D}$ be the set of all nodes of G designated by Algorithm 1 as dominators. In the next few lemmas we will discuss some properties of Algorithm 1 and of the associated dominating set $\mathcal{D}$. We will conclude with the main Theorem 1. The first lemma shows that the algorithm is local in the sense that it terminates in constant time and the actions of nodes in a hexagon are influenced only

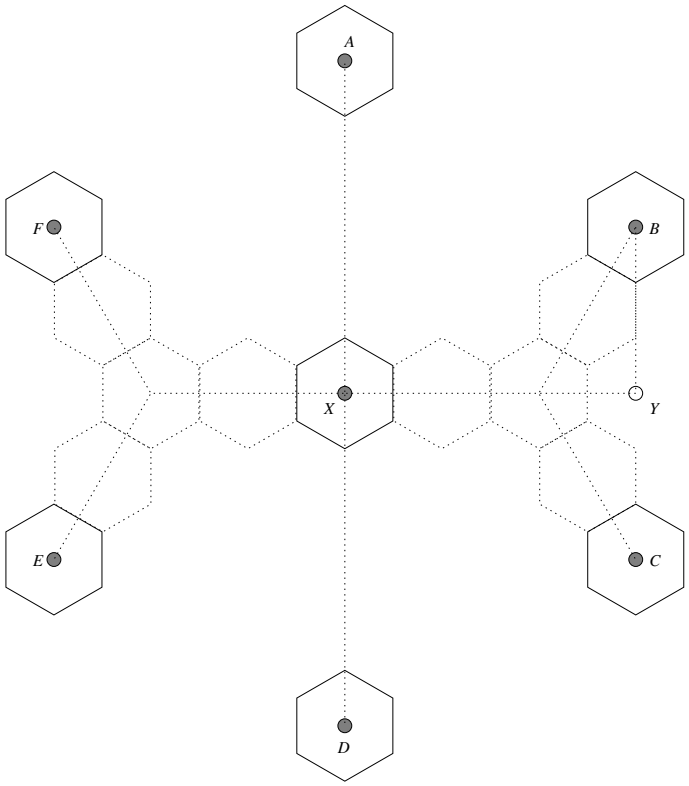

Figure 4: Six hexagons A, B, C, D, E, F (indicated with a gray bullet at their center) "adjacent" to a given hexagon X (also indicated with a gray bullet at its center), all in the same class. It is clear from the picture that $|XY| = 6\sqrt{3}/4$ and $|YB| = 3/2$.

by actions of nodes a constant number of hops away.

Lemma 2 *The selection of a dominator in a hexagon of Class i by Algorithm 1 depends only on information received from nodes that are at most $i-1$ hops away from nodes in the given hexagon.*

Proof. The simplest way to see the claim is using induction on i. To begin, it is clear that the selection of a dominator of nodes in a hexagon of class 1 is done by examining only the nodes inside this hexagon. Let k be an integer greater than 1 and assume that for any $j < k$ the calculation of a dominator in a hexagon of class j depends on information from nodes that are at distance at most $j-1$ hops from the given hexagon. The selection of a dominator in a hexagon of class k in Step 5 is done after nodes of class $j < k$ that are one hop away complete their calculations, and by assumption this requires at most $k-1$ additional hops. In Figure 5 we depict as shaded all hexagons that are involved in calculating a dominator of a hexagon of class 6. This completes the proof of Lemma 2. ■

Lemmas 3 and 4 are concerned with domination and independence of the resulting set of vertices selected by the algorithm.

Lemma 3 *Every vertex of G is either in $\mathcal{D}$ or adjacent to a vertex in $\mathcal{D}$. Thus, set $\mathcal{D}$ is a dominating set of G.*

Proof. Let X be a node of G that is not in $\mathcal{D}$. If X is of class 1, then one of the nodes in the hexagon containing X is designated as a dominator in Step 3. Since the diameter of a hexagon is

Algorithm 1 Local Dominating Set Algorithm

// Algorithm is executed independently by each node. Execution starts either
// when a node needs to find its dominator, or if it receives a request to find
// a dominator in its hexagon.

1: Determine your class number using your coordinates and the tiling information.
2: Find all your neighbors and obtain their coordinates and class numbers.
3: If your class number is 1 then the node N in your hexagon closest to the center of it is designated as a dominator. Continue to Step 6.
4: Find whether there is a node in your hexagon that has no neighbor of lower class. If such nodes exist then the one of them that is closest to the center of the hexagon is designated as a dominator. Continue to Step 6.
5: If you have a neighbor M of a lower class number then send to M a request to execute the algorithm for finding its dominator. Once the replies from all neighbors of lower class number are received, determine if you are already dominated. Inform your neighbors in your hexagon of the result. When all nodes of your hexagon finish this calculation, node N in your hexagon closest to the center and not dominated yet is designated as a dominator, if such a node exists.
6: Inform all your neighbors that a dominator selection in your hexagon is completed and give them its result.
7: Terminate your execution of the algorithm.

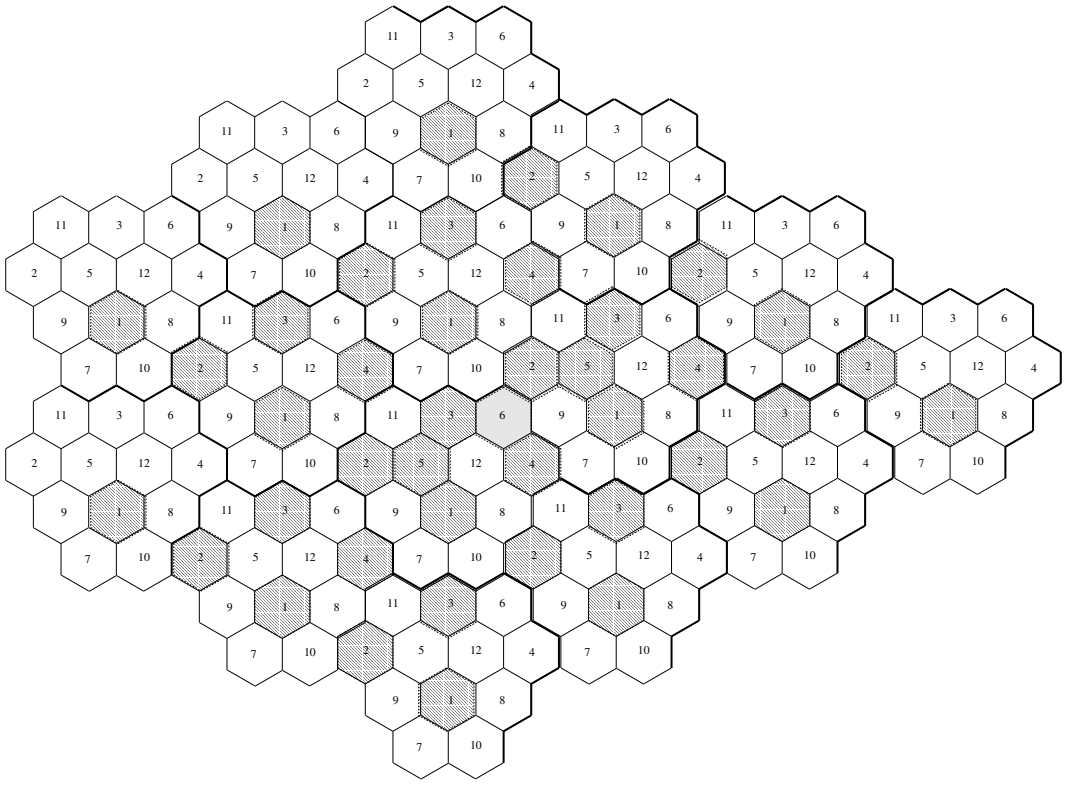

Figure 5: Hexagons involved in calculating a dominator of Class 6.

1, node X is dominated by the designated node. If X is of class $i > 1$ and at least one node of its hexagon is not dominated by a node of an adjacent lower class, one of the nodes in this hexagon is

designated as a dominator in Step 4 of the algorithm. Since the diameter of a hexagon is 1, node X is dominated by the designated node. This completes the proof of Lemma 3. ■

Lemma 4 *The Euclidean distance between any two nodes of $\mathcal{D}$ is more than one. Thus $\mathcal{D}$ is an independent set of G.*

Proof. Every hexagon contains at most one element of $\mathcal{D}$. According to Lemma 1, the distance between any two vertices of different hexagons of the same class is greater than 2. Thus, the dominators selected in hexagons of class i for any fixed i are pairwise independent. In a hexagon of class $i > 1$, a dominator is designated in Step 4 of the algorithm only as a vertex that is not dominated by an adjacent element in $\mathcal{D}$ of lower class. Thus the distance of a dominator belonging to a class i to dominators of class $j < i$ is more than 1. This completes the proof of Lemma 4. ■

We now summarize the properties of the dominating set calculated by Algorithm 1 in the following theorem.

Theorem 1 *Let G be a unit disk graph and $\mathcal{D}$ be the set of dominators calculated by Algorithm 1. $\mathcal{D}$ is a dominating, independent set of G and for any dominating set $\mathcal{D}^*$ of G, we have $|D|/|D^*| \leq 5$. Thus the competitive ratio of Algorithm 1 is 5.*

Proof. According to Lemmas 3 and 4, $\mathcal{D}$ is both a dominating and an independent set of G. Let $\mathcal{D}^*$ be a minimum dominating set of G and N be a vertex of $\mathcal{D}^*$. Consider the elements of $\mathcal{D}$ that are dominated by N, i.e., the nodes of $\mathcal{D}$ that lie in the circle of radius 1 around N. If one of the elements of $\mathcal{D}$ is equal to N then by Lemma 4 no other element of $\mathcal{D}$ can be in this circle. If no element of $\mathcal{D}$ is equal to N and there are 6 or more such elements of $\mathcal{D}$, then there are at least two elements, say X_1 and X_2, such that the angle $\angle X_1 N X_2$ is at most $\pi/3$. However this implies that the distance between X_1 and X_2 is at most 1, a contradiction to Lemma 4. Thus we can conclude that an element of $\mathcal{D}^*$ dominates at most 5 elements of $\mathcal{D}$. This implies that $|D|/|D^*| \leq 5$. This completes the proof of Theorem 1. ■

2.2 Lower bound example

Figure 6 depicts an example of a set of points so that the ratio between the minimum dominating set and the dominating set $\mathcal{D}$ calculated by Algorithm 1 is greater than 4. If the vertices of degree 5 are placed in hexagons of high class number, the dominating set $\mathcal{D}$ computed by the algorithm includes only the vertices of degree 1 and 2 while the optimum one consists of the vertices of degree 5.

3 Local Algorithm for Connected Dominating Set of a UDG

Our construction of a connected dominating set starts with the dominating set constructed by the algorithm of Section 2 and we make it connected by adding selected vertices, called *bridges*. We first need to introduce some notation and study properties of optimal connected dominating sets.

Let OCDS denote an *optimal connected dominating set*, i.e. the minimal subset of vertices which induces a connected subgraph of a given unit disk graph G and such that any vertex of G has a neighbor in OCDS. Consider a Minimum Spanning Tree T of OCDS, i.e. a connected subgraph of G containing the OCDS and such that it has the smallest possible sum of lengths of its edges. Note that the minimal angle between any two incident edges (x, y) and (x, z) of T is $\pi/3$, otherwise edge (y, z) would be used in T instead of (x, y) or (x, z). Suppose that T is rooted at some vertex r, of degree smaller than six. We denote by D_v the disk of radius 1 centered at v. Consider the dominating set $\mathcal{D}$ constructed in Section 2. We will find an upper bound on the number of vertices of $\mathcal{D}$. For this purpose we will charge to vertices of T small subsets of $\mathcal{D}$ as follows. Suppose $(u, v) \in T$,

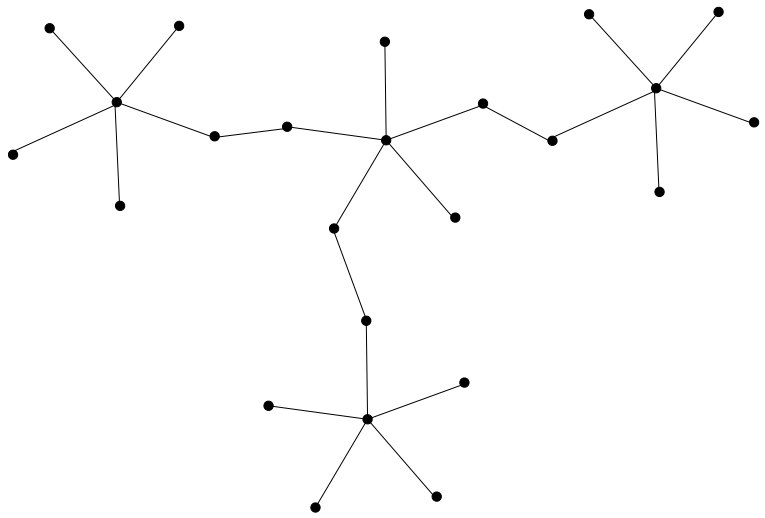

Figure 6: Graph realizing competitive ratio greater than 4

i.e. u is the parent of v in T. We charge to v all vertices centered inside D_v except those which are inside D_u. Hence the number d_v of vertices charged to v is $d_v = |\mathcal{D} \cap (D_v \setminus D_u)|$. Note that, since T contains a dominating set of the UDG, each vertex of $\mathcal{D}$ is charged to at least one vertex of T.

Furthermore, let E_v be the set of children of v, such that there is no vertex of $\mathcal{D}$ reachable from both v and a vertex of E_v, i.e., $E_v = |\{w : w \text{ is a child of } v \text{ in } T \text{ and } \mathcal{D} \cap (D_v \cap D_w) = \emptyset\}|$. Denote $e_v = |E_v|$.

Lemma 5 *Let T be a Minimum Spanning Tree with root r of an optimal connected dominating set of UDG G with d_v, e_v defined for any vertex v of T as above.*

1. *If v is different from r then $d_v + e_v \leq 4$.*
2. *$d_r + e_r \leq 5$.*

Proof. To prove part 1 of the lemma, let us first consider the vertices of $\mathcal{D}$ inside D_v, counted in d_v. Firstly, vertices from $D_u \cap D_v$, are charged to d_u, hence a circular sector of angular size at least $2\pi/3$ of D_v does not contain any vertex of $\mathcal{D}$ counted in d_v (see Figure 7). As $\mathcal{D}$ is an independent set, for any $x, y \in \mathcal{D} \cap D_v$, the angle $\angle xvy > \pi/3$. Similarly, since angles between edges in T are greater or equal to $\pi/3$, for any $x, y \in E_v$, the angle $\angle xvy \geq \pi/3$. Finally, by definition of E_v, for any $x \in \mathcal{D}, y \in E_v$, the angle $\angle xvy > \pi/3$. Since in the angular region around v of size smaller than $2\pi/3$ we can place at most four points such that any two of them, say x, y, form an angle $\angle xvy > \pi/3$ we have $d_v + e_v \leq 4$.

To prove part 2 of the lemma, we similarly observe that $d_r + e_r \leq 5$. Indeed, since $e_r \leq 5$, in case $d_r + e_r > 5$ the sum of angles formed by the vertices from $(\mathcal{D} \cup E_r) \cap D_r$ would be strictly greater than 2π. ■

Corollary 1 *Let f be the number of edges $\{u, v\}$ in T such that $\mathcal{D} \cap D_u \cap D_v \neq \emptyset$. Then $|\mathcal{D}| \leq 3|OCDS| + 2 + f$.*

Proof. Indeed, from the definition of f and e_v we derive that

$$|OCDS| - 1 - f = \sum_{v \in OCDS} e_v.$$

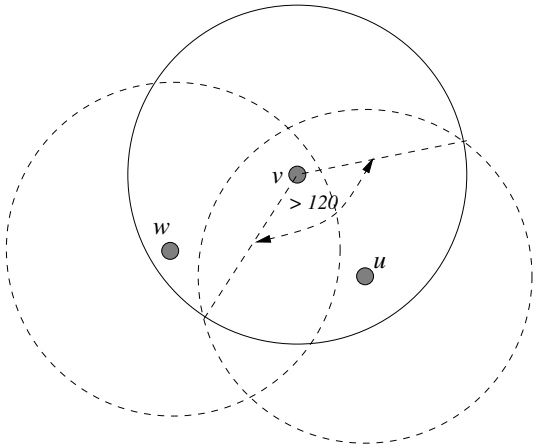

Figure 7: Adjacent vertices of UDG have a common circular sector of size at least $2\pi/3$ of the unit disk.

Summing over all $v \in OCDS$ we get

$$\begin{aligned}
|\mathcal{D}| &= \sum_{v \in OCDS} d_v \\
&= d_r + \sum_{v \in OCDS \setminus \{r\}} d_v \\
&\leq 1 + \sum_{v \in OCDS} (4 - e_v) \\
&= 4 \cdot |OCDS| + 1 - \sum_{v \in OCDS} e_v \\
&= 4 \cdot |OCDS| + 1 - (|OCDS| - 1 - f) \\
&= 3 \cdot |OCDS| + 2 + f
\end{aligned}$$

This completes the proof of the corollary. ■

Note that the competitive ratio of $\mathcal{D}$ with respect to the OCDS is better than the one from Theorem 1 since f can be at most $|OCDS| - 1$.

Just like the dominating set previously considered in Section 2, we consider a tiling of the plane with tiles. Each tile consists of c hexagons of radius 1 that are being assigned different class numbers and such that hexagons of the same class number are at distance at least k from each other. The dominating set in Section 2, required tiles with $c = 12$ hexagons in order to achieve distance $k = 2$ (see Lemma 1). The tiles we consider in this section achieve distances k bigger than 2 and it can be proved easily that in this case the required class number c is in $\Theta(k^2)$.

Intuitively, the algorithm for constructing a connected dominating set is as follows. Find a dominating set using Algorithm 1 and select a coordinator vertex in each non-empty hexagon using some election procedure. The coordinators are responsible for augmenting the existing dominating set by adding bridges, each bridge (one or two vertices) joining at least two connected components. We suppose that each vertex will communicate information at distance less than k hops from it. Each hexagon is assigned a class number (from 1 to c) so that vertices of two hexagons of the same class number are at least at distance k from each other.

The algorithm consists of two phases, each of which consists of c rounds. In the first phase, the algorithm repeatedly inserts single vertex bridges, while in the second phase, the double vertex

bridges are added, eventually resulting in the set becoming connected. In round i, $1 \leq i \leq c$, coordinators from hexagons of class i will act, by trying to add connecting bridges within their hexagons. We suppose that each coordinator takes a decision about adding a bridge based on its knowledge of the part of the UDG at distance less than k from it. Hence, when a coordinator decides to add a bridge it is possible that the bridge is not joining two components from global perspective but rather two components as perceived from the local, limited perspective of the coordinator, and the resulting graph thus may contain some cycles. Algorithm 2 which is described below is a more formal outline of this idea.

Algorithm 2 Local Algorithm for Connected Dominating Set

1: Compute the dominating set $\mathcal{D}$ applying Algorithm 1.
2: Select a coordinator vertex in each non-empty hexagon and determine its colour c based on the coordinates.
// The rest of the algorithm is specified for a hexagon H.
3: Set the local set of selected vertices S_H to be the vertices of $\mathcal{D}$ at distance less than k hops from H (i.e. each vertex of $\mathcal{D}$ is broadcasted up to distance k).
4: **for** bridgesize= 1 to 2 **do**
5: **for** round= 1 to c **do**
6: **if** $class(H) = round$ **then**
7: Determine all connected components of the UDG induced by the vertices of S_H.
8: **repeat**
9: Find a set B of vertices of H, such that $|B| = bridgesize$ and such that adding B to S_H connects at least two different connected components.
10: Add B to S_H and locally recompute the connected components.
11: **until** no such set B can be found
12: Broadcast the newly added vertices up to distance k hops.
// Hexagon H has done its job for this bridgesize, now it only participates in the broadcasts and updates its S_H.
13: **else**
14: **for** each non-empty hexagon H' at distance at most k **do**
15: wait for a message containing the vertices V'_H added in H'.
16: $S_H \leftarrow S_H \cup V'_H$
17: **end for**
18: **end if**
19: **end for**
20: **end for**
21: The union of S_H for all hexagons H is the desired connected dominating set $\mathcal{S}$.

The complexity analysis of this algorithm follows in the sequel. However, note that previous "global algorithms" made use of a "globally constructed spanning tree" in order to find one-node bridges. We can no longer use a global algorithm to ensure that all our bridges consist of a single node. Rather we can use the structure of the dominating sets to show that if we first add all the single node bridges then we can limit the number of the remaining two-node bridges not just by the size of the dominating set, but rather by the size of the connected dominating set.

Lemma 6 *Consider the set $\mathcal{S}' = \bigcup_{H \in \mathcal{H}} S'_H$, where S'_H is the set of selected vertices of hexagon H after the first phase, and $\mathcal{H}$ is the set of all non-empty hexagons. Consider the MST T of the OCDS used in Lemma 5. Let f be the number of edges $\{u, v\}$ of T such that $\mathcal{D} \cap (D_u \cap D_v) \neq \emptyset$. Then the UDG of $\mathcal{S}'$ has at most $|OCDS| - f$ connected components.*

Proof. Consider the disk D_v of radius 1 centered at a vertex v of the OCDS. After the first phase, all vertices of $\mathcal{D} \cap D_v$ must belong to the same component. Indeed, at some time the hexagon H of

v would have been considered – if those vertices were not in the same component at the time v was considered on line 12, v would have been added to S_H, connecting them all. This means that there may be at most $|OCDS|$ components, each corresponding to a vertex of OCDS. However, for each edge $\{u, v\}$ counted in f it follows that the components of u and v are connected. ■

Lemma 7 *$\mathcal{S}$ as constructed by Algorithm 2 is a connected dominating set.*

Proof. Since by construction $\mathcal{D} \subseteq \mathcal{S}$ and $\mathcal{D}$ is a dominating set, it is sufficient to prove that $\mathcal{S}$ is connected. Suppose, to the contrary, that it has at least two components, say $\mathcal{S}_1$ and $\mathcal{S}_2$. Since the original UDG G is connected, there must be a path in G connecting $\mathcal{S}_1$ and $\mathcal{S}_2$. Consider the shortest such path, say π. Colour vertices of this path white if they are adjacent to a dominating vertex from $\mathcal{S}_1$, and black otherwise. As $\mathcal{S}$ is dominating set, each vertex of π will be coloured. Since the endpoints of π are of different colour, there must be an edge $\{u, v\}$ such that u and v are of different colour. However, at some moment the algorithm would have considered (at line 12 or 15) adding one or both of these vertices, thus connecting $\mathcal{S}_1$ and $\mathcal{S}_2$. ■

We will also use the following lemma from Funke et al. [2006]:

Lemma 8 *The size of any independent set in a unit disk graph G is at most $3.453 \cdot |OCDS| + 8.291$*

We are now ready to prove the main theorem of this section.

Theorem 2 *Let $k > 3$ be an integer and $\mathcal{S}$ be the connected dominating set computed by Algorithm 2 with parameter k. Then*

$$|\mathcal{S}| \leq \left(7.453 + \frac{15}{k-3}\right) \cdot |OCDS| + \frac{16.6}{k-3}. \tag{1}$$

Proof. We can view $\mathcal{S}$ as obtained from $\mathcal{D}$ by sequentially adding vertices (or pairs of vertices):

1. the vertices are added in the order according to the class number of the hexagon that contained them,
2. within each hexagon the vertices are added in the order they were considered by the algorithm, and
3. the order of the vertices added by different hexagons of the same class number does not matter.

Consequently, we have a sequence $\mathcal{D} = \mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_r = \mathcal{S}$. For each $\mathcal{S}_i$ we can define VG_i as the graph obtained by applying the Local Minimal Spanning Tree construction of Chávez et al. [2006] to the UDG of $\mathcal{S}_i$. Since different hexagons of the same color are at least at distance k away from each other, each graph VG_i is a planar spanner of $\mathcal{S}_i$ and each of its faces is of size at least $2k$. We will denote by VG the graph obtained by considering the final $\mathcal{S} := \mathcal{S}_r$.

We say that an addition of a node (or a pair of nodes) to obtain $\mathcal{S}_i$ from $\mathcal{S}_{i-1}$ is *idle* if the number of connected components in VG_i is equal to the number of components in VG_{i-1}. Let us denote by V_0 the size of $\mathcal{D}$, V_1' and V_2' be the numbers of additions of single nodes and pairs of nodes, respectively, which are not idle. Similarly, let V_1'' and V_2'' be the numbers of single and double-node idle additions, respectively. Let V denote the number of nodes of VG (i.e. $V = |\mathcal{S}|$), E denote the number of edges and F the number of faces. From Euler's formula we have that

$$F + V = E + 2. \tag{2}$$

Therefore, from the way $\mathcal{S}$ was constructed, we have

$$V = V_0 + V_1' + V_1'' + 2V_2' + 2V_2''. \tag{3}$$

Note that our algorithm adds a vertex (or pair of vertices) only when they connect at least two connected components (when looking up to distance $k-1$). Therefore, only two or three edges are

added to VG_{i-1} in order to get VG_i (depending on whether a single vertex bridge or a two-vertex bridge has been added). Combining this observation with the definition of idle additions, we have

$$E \geq V - 1 + V_1'' + V_2''. \tag{4}$$

As initially there were V_0 components (recall that $\mathcal{D}$ is an independent set) and at the end VG is connected, from the definitions of V_1' and V_2' we have

$$V_1' + V_2' \leq V_0 - 1. \tag{5}$$

By Lemma 6

$$V_2' \leq |OCDS| - f - 1. \tag{6}$$

Finally, as each face is of size at least $2k$ and the sum of face sizes is exactly $2E$ we get

$$kF \leq E. \tag{7}$$

Substituting for F into Identity 2 according to Inequality 7 we get $E/k + V \geq E + 2$ and therefore $V - 2 \geq E(1 - 1/k)$. After substituting for E according to Inequality 4 and using some elementary calculations we get

$$\frac{V}{k-1} - 1 \geq V_1'' + V_2''. \tag{8}$$

Substituting all of this into Equality 3 we get

$$\begin{aligned} V &\leq V_0 + V_1' + V_2' + V_2' + 2V_1' + 2V_2'' \\ &\leq V_0 + V_0 - 1 + |OCDS| - f + \frac{2V}{k-1} - 2. \end{aligned}$$

Using Corollary 1 we obtain

$$\begin{aligned} V &\leq 2V_0 - 3 + |OCDS| - f + \frac{2V}{k-1} \\ &\leq V_0 - 1 + 4|OCDS| + \frac{2V}{k-1} + 1. \end{aligned}$$

Hence, by Lemmas 4 and 8 we get

$$V \leq 7.453 \cdot |OCDS| + \frac{2V}{k-1} + 8.291, \tag{9}$$

which yields

$$\begin{aligned} |\mathcal{S}| &= V \\ &\leq \frac{7.453 \cdot |OCDS| + 8.291}{1 - \frac{2}{k-1}} \\ &\leq \left(7.453 + \frac{15}{k-3}\right) \cdot |OCDS| + \frac{16.6}{k-3} \end{aligned}$$

and concludes the proof of the theorem. ■

Algorithm 2 does not produce an optimal connected dominating set of a given graph. It is rather obvious that a strictly local algorithm cannot produce an optimal connected dominating set even for a long cycle. In some cases we can lower the number of vertices in our connected dominating set by adding one more phase to Algorithm 2: after the addition of bridges is finished we can check for each vertex of D whether it is still needed for domination, following the order of the class numbers, and remove it from the connected dominating set if it is not needed for domination and it does not create a disconnection using the neighbourhood of size $k - 1$. However, this does not improve the competitive ratio of Theorem 2.

4 Conclusion

This paper gives the first ever local algorithms for constructing dominating and connected dominating sets of unit disk graphs with location aware nodes. The competitive ratios are 5 and $7.453 + \epsilon$, respectively. Although it was shown that the competitive ratio of first algorithm could not be improved further, we do not have a lower bound on the competiive ratio of the second algorithm. This also poses a rather fundamental question about the "power of geometric awareness" and its impact on distributed computing, which is expected to have repercussions on future research studies on this subject.

References

K. M. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected-dominating-set construction for routing in mobile ad hoc networks. In *MOBIHOC '02*, pages 157–164, 2002.

J. Aspnes, C. Bush, S. Dolev, P. Fatouroum, C. Georgiou, A. Shvartsman, P. Spirakis, and R. Wattenhofer. Eight open problems in distributed computing. *Bulletin of the European Association for Theoretical Computer Science*, 90:109, Oct. 2006. Columns: Distributed Computing.

Awerbuch and Peleg. Sparse partitions (extended abstract). In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1990.

H. Breu and D. G. Kirkpatric. Unit disk graph recognition is NP-hard. *Computational Geometry: Theory and Applications*, 9, 1998.

E. Chávez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia. Local construction of planar spanners in unit disk graphs with irregular transmission ranges. In J. R. Correa, A. Hevia, and M. A. Kiwi, editors, *LATIN*, volume 3887 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 2006.

B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86: 165–177, 1990.

A. Dessmark and A. Pelc. Broadcasting in geometric radio networks. *Journal of Discrete Algorithms*, 2007. in press.

F. Fich and E. Ruppert. Hundreds of impossibility results for distributed computing. *Distributed Computing*, 16, 2003.

S. Funke, A. Kesselman, U. Meyer, and M. Segal. A simple improved distributed algorithm for minimum cds in unit disk graphs. *ACM Transactions on Sensor Networks*, 2(3):444–453, August 2006.

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. *Journal of Algorithms*, 26(2):238–274, Feb. 1998.

R. W. Irving. On approximating the minimum independent dominating set. *Information Processing Letters*, 37, 1991.

L. Jia, R. Rajaraman, and R. Suel. An efficient distributed algorithm for constructing small dominating sets. *Distributed Computing*, 15, 2002.

D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9, 1974.

F. Kuhn and R. Wattenhofer. Constant-time distributed dominating set approximation. *Distributed Computing*, 17(4):303–310, 2005.

F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *23th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 2004a.

F. Kuhn, T. Moscibroda, and R. Wattenhofer. Unit disk graph approximation. In *DialM: Proceedings of the Discrete Algorithms and Methods for Mobile Computing & Communications; later DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, 2004b.

F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Local approximation schemes for ad hoc and sensor networks. In *DialM: Proceedings of the Discrete Algorithms and Methods for Mobile Computing & Communications; later DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, 2005a.

F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the locality of bounded growth. In *24th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 2005b.

F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. In *SODA*, pages 980–989. ACM Press, 2006.

N. Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, Feb. 1992.

M. Luby. A simple parallel algorithm for the maximal independent set problem. In *Proc. 17th Annual ACM Symposium on Theory of Computing (STOCS)*, pages 1–10, May 1985.

C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994.

M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Geometry based heuristics for unit disk graphs, 1994. 19 pages.

M. Naor and L. Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24, 1995.

T. Nieberg and J. Hurink. A ptas for the minimum dominating set problem in unit disk graphs. In *Proc.3rd Workshop on Approximation and Online (WAOA 2005)*, 2005.

D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.