Energy Conservation via Domatic Partitions

Sriram V. Pemmaraju
Department of Computer Science
University of Iowa
Iowa City, IA 52242
sriram@cs.uiowa.edu

Imran A. Pirwani
Department of Computer Science
University of Iowa
Iowa City, IA 52242
pirwani@cs.uiowa.edu

ABSTRACT

Using a dominating set as a coordinator in wireless networks has been proposed in many papers as an energy conservation technique. Since the nodes in a dominating set have the extra burden of coordination, energy resources in such nodes will drain out more quickly than in other nodes. To maximize the lifetime of nodes in the network, it has been proposed that the role of coordinators be rotated among the nodes in the network. One abstraction that has been considered for the problem of picking a collection of coordinators and cycling through them, is the *domatic partition problem*. This is the problem of partitioning the set of the nodes of the network into dominating sets with the aim of maximizing the number of dominating sets. In this paper, we consider the *k-domatic partition problem*. A *k-dominating set* is a subset D of nodes such that every node in the network is at distance at most k from D. The *k-domatic partition problem* seeks to partition the network into maximum number of k-dominating sets. We point out that from the point of view of saving energy, it may be better to construct a k-domatic partition for $k > 1$.

We present three deterministic, distributed algorithms for finding large k-domatic partitions for $k > 1$. Each of our algorithms constructs a k-domatic partition of size at least a constant fraction of the largest possible $(k-1)$-domatic partition. Our first algorithm runs in constant time on unit ball graphs (UBGs) in Euclidean space assuming that all nodes know their positions in a global coordinate system. Our second algorithm drops knowledge of global coordinates and instead assumes that pairwise distances between neighboring nodes are known. This algorithm runs in $O(\log^* n)$ time on UBGs in a metric space with constant doubling dimension. Our third algorithm drops all reliance on geometric information, using connectivity information only. This algorithm runs in $O(\log \Delta \cdot \log^* n)$ time on growth-bounded graphs. Euclidean UBGs, UBGs in metric spaces with constant doubling dimension, and growth-bounded graphs are successively more general models of wireless networks and all three models include the well-known, but somewhat simplistic wireless network models such as unit disk graphs.

MobiHoc'06, May 22–25, 2006, Florence, Italy.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; C.2.4 [**Computer Systems Organization**]: computer-communication networks—*Distributed Systems*

General Terms

Algorithms, Theory.

Keywords

Distributed algorithms, domatic partition, dominating sets, doubling dimension, growth-bounded graphs, maximal independent sets, metric space, network decomposition, unit disk graphs, unit ball graphs.

1. INTRODUCTION

A wireless sensor network consists of individual nodes that are able to sense their environment (sensor), communicate with nearby nodes via radio broadcast (network), and perform local computations based on information gathered from the surroundings. Once deployed, a sensor network may not permit regular maintenance. This may be due to a variety of reasons: the network may consist of a very large number of nodes or the nodes may be in an environment in which regular human intervention is either impossible or undesirable [11, 17]. Nodes in a sensor network come equipped with battery and from the point of deployment, this battery reserve becomes a valuable resource since it cannot be replenished. Hence, maximizing the lifetime of the network by minimizing the energy consumption is an important challenge in wireless sensor networks.

A standard approach for reducing energy consumption is to carefully schedule node activity. As has been observed in [4], whenever there are sufficiently many nodes in a region, only a small fraction of nodes need be active for forwarding messages, etc. The rest of the nodes can enter a *sleep* mode, thereby conserving energy. The problem of maximizing the number of nodes that are asleep at any given time while maintaining sufficient activity in the network is usually modeled as the problem of finding a small *dominating*

set in the network. Once a small dominating set is found, the nodes in the dominating set collectively act as "coordinators" for the network and the rest of the nodes go to sleep. To maximize the lifetime of the network it is critical that the role of coordinators be rotated among the nodes in the network, so that every node gets a chance to sleep. This issue has been considered in [4, 19]. In [4], a distributed, randomized algorithm called *span* is presented, in which nodes make local decisions to sleep or to join the set of coordinators and nodes in the network take turns at being coordinators. In [19], the problem of rotating the responsibility of being a coordinator has been abstracted as the *domatic partition problem*.

We start by assuming that all nodes in the network have identical transmission ranges and this allows us to model the network as an undirected graph $G = (V, E)$ where the vertex set represents the nodes and each edge in the edge set represents two nodes that are within each other's transmission range. A *dominating set* $D \subseteq V$ of G is a subset of vertices such that each $v \in V$ is either in D or has a neighbor $u \in D$. A *domatic partition* is a partition $\mathcal{D} = \{D_1, D_2, \ldots, D_t\}$ of V such that each block D_i of $\mathcal{D}$ is a dominating set of G. Suppose that $\mathcal{D} = \{D_1, D_2, \ldots, D_t\}$ is a domatic partition of G. Then a simple schedule for the nodes would be for the nodes in D_1 to be active for some fixed period of time T, during which the rest of the nodes are asleep, followed by a period of time T in which nodes in D_2 are active, while the rest of the nodes are asleep, and so on. Such a schedule would imply that in the long run, each node is active for roughly $1/t$ of the time. Therefore maximizing t leads to minimizing this fraction. This motivates the *domatic partition problem*:

`DOMATIC PARTITION`
INSTANCE: A graph $G = (V, E)$
OUTPUT: A partition $\mathcal{D} = \{D_1, D_2, \ldots, D_t\}$ of V of maximum size such that each D_i is a dominating set of G.

The schedule suggested above may be somewhat simplistic because it does not pay attention to possible differences in the amount of energy available at different nodes. However, the more general problem in which nodes start of with different battery supply does not seem much harder than the "uniform" version of the problem in which nodes are assumed to be identical (see [19] for example). We only consider the uniform version of the problem here.

1.1 Results

In this paper we present fast, deterministic, distributed algorithms for finding large *k-domatic partitions* in various graph models of wireless sensor networks. Let $d(u, v)$ denote the length of a shortest uv-path in G measured by counting the number of edges in the path. For any integer $k \geq 1$, let $N_k(v) = \{u : 0 < d(u, v) \leq k\}$. We call $N_k(v)$ the *k-neighborhood* of v and any vertex $u \in N_k(v)$ is called a *k-neighbor* of v. A *k–dominating set* $D^{(k)} \subseteq V$ of G is a vertex set such that each $v \in V$ is either in $D^{(k)}$, or has a k-neighbor in $D^{(k)}$. A *k-domatic partition* is a partition $\mathcal{D} = \{D_1, D_2, \ldots, D_t\}$ of V such that each block D_i of $\mathcal{D}$ is a k-dominating set of G. Note that a 1-domatic partition is just a domatic partition.

We consider k-domatic partitions rather than domatic partitions simply because as k increases, we expect the size of a largest k-domatic partition to also increase. For example, a network may have a much larger 2-domatic partition than the largest 1-domatic partition and from the point of view of scheduling this implies that each node can stay asleep for a much larger fraction of time. However, this advantage may be somewhat offset by the fact that it may take 2 hops for a node not in a 2-dominating set to reach a node in the 2-dominating set. To be more specific, suppose that $D^{(2)}$ is a 2-dominating set of G and further suppose that a node $u \in V - D^{(2)}$ needs to send information to some node in $D^{(2)}$. This is problematic because there may be no node of $D^{(2)}$ in u's neighborhood. One solution to this problem is to increase u's transmission range so that some node in $D^{(2)}$ is in its range. The fact that some node in $D^{(2)}$ is a 2-neighbor of u implies that u does not have to increase its transmission range too much to reach $D^{(2)}$. So the advantage we gain by having a 2-domatic partition of larger size may be offset to some extend by the fact it costs nodes more energy to communicate with the set of coordinators. In some situations we can control the amount of extra energy needed by a node to reach $D^{(2)}$. For example, one of the algorithms we present (Algorithm k-`DP1` in Section 3.1) can be made to take an additional parameter $\epsilon > 0$ such that for each vertex $v \in V$ and for each $D^{(k)}$ in the k-domatic partition constructed by the algorithm, v can reach a node in $D^{(k)}$ by increasing its transmission range by at most ϵ. We do not explore this trade-off any further in this paper, merely noting that results from our experiments indicate that for small values of k, the size of an optimal k-domatic partition grows substantially as k increases. These preliminary results indicate that an optimal value of k may be some small value larger than 1.

We present algorithms for computing, for any $k \geq 2$, a k-domatic partition whose size is within a constant fraction of the size of an optimal $(k-1)$-domatic partition. To the best of our knowledge, such size guarantees have never been provided for k-domatic partitions. To state this more precisely, we need some definitions. Let δ denote the smallest vertex degree in a graph. More generally, for any integer $k \geq 1$, let $\delta_k = \min_v\{|N_k(v)|\}$. For any k-domatic partition $\mathcal{D}$ of a graph, $|\mathcal{D}| \leq \delta_k + 1$. This is because for every vertex v, every block in $\mathcal{D}$ contains at least one vertex in $\{v\} \cup N_k(v)$. Thus, the size of an optimal k-domatic partition is bounded above by $\delta_k + 1$. We present three algorithms (on different classes of graphs and with different running times) that compute a k-domatic partition of size at least $(\delta_{k-1} + 1)/c_k$ for some constant c_k. Here c_k is a constant in the sense that it does not depend on graph parameters such as the number of vertices, diameter, or maximum degree. However, c_k may depend on k, as the subscript indicates. Since $\delta_k + 1$ is an upper bound on the size of any k-domatic partition, if we had shown that the size of the computed k-domatic partition is at least $(\delta_k + 1)/c_k$, then that would have been a $\frac{1}{c_k}$-factor approximation algorithm. For any fixed k, this would be an $O(1)$-factor approximation. However, we are unable to show this and instead show a weaker lower bound (namely, $(\delta_{k-1}+1)/c_k$) on the size of the k-domatic partition returned by our algorithm. In fact, for the classes of graphs we consider, it is unknown whether there is always a k-domatic partition of size at least some constant fraction of $(\delta_k + 1)$. Our result should be contrasted with the current state of affairs, which is that even for the 1-domatic partition problem on UDGs, the best known approximation algorithm [6, 19] returns a partition of size at least $(\delta + 1)/O(\log \Delta)$. This algorithm also solves the k-domatic partition problem and

returns a partition of size at least $(\delta_k+1)/O(\log\Delta)$. In general, this lower bound and the lower bound of $(\delta_{k-1}+1)/c_k$ that we obtain, are incomparable, but for certain classes of graphs our lower bound is much better.

The most general class of graphs that our results apply to is the class of *growth-bounded graphs* [14]. A graph G is *f-growth-bounded* if there is a function f on non-negative integers such that for every integer $r \geq 1$, every r-neighborhood in G contains an independent set of size at most $f(r)$. The critical aspect of this definition is that the size of a largest independent set in an r-neighborhood depends only on r and not on any other graph parameters. This means that for any fixed r, the size of a largest independent set in any r-neighborhood is bounded above by a constant. Well-known graph models of wireless networks such as *unit disk graphs* (UDGs) and *quasi unit disk graphs* (qUDGs) are both subclasses of growth-bounded graphs. Recall that a *UDG* is a graph $G=(V,E)$ whose vertex set V can be placed in one-one correspondence with a set of points in the Euclidean plane and whose edges connect exactly those pairs of vertices u, v whose Euclidean distance $|uv|$ is at most one. For any fixed α, $0<\alpha<1$, an *α-qUDG* is a graph $G=(V,E)$ whose vertex set V can be placed in one-one correspondence with a set of points in the Euclidean plane and whose edge set E satisfies the constraint: if $|uv|\leq\alpha$ then $\{u,v\}\in E$ and if $|uv|>1$ then $\{u,v\}\notin E$. The qUDG model does not say whether a pair of vertices whose distance is in the range $(\alpha,1]$ are to be connected by an edge or not. By leaving this unspecified, this model attempts to take into account transmission errors, fading signal strength, and physical obstructions which cause real transmission ranges to not be shaped like disks. In general, growth-bounded graphs capture in a simple way the fairly intuitive geometric property of wireless networks that if many nodes are close to each other, they will tend to hear each others' transmissions and therefore only a small number of these can be mutually independent.

We provide faster algorithms for a certain subclass of growth-bounded graphs. A *unit ball graph* (UBG) is a graph $G=(V,E)$ whose vertices reside in some metric space and whose edges connect pairs of vertices whose distance is at most one. The *doubling dimension* of a metric space is the smallest ρ such that any ball in this metric space can be covered by 2^ρ balls of half the radius. A metric space with constant doubling dimension is called a *doubling metric*. We call a UBG a *doubling UBG* if it resides in a doubling metric space. It is easy to see that for any fixed d, the d-dimensional Euclidean space is a doubling metric. Thus a UBG in fixed dimensional Euclidean space is a doubling UBG. An arbitrary UBG need not be growth-bounded, but any doubling UBG is indeed growth-bounded (Lemma 1 in [14]). Taking advantage of the geometry of a UBG and the fact that pairwise distances are available, we provide a faster algorithm for computing large k-domatic partitions on doubling UBGs. Our interest in doubling UBGs is due to the fact that they significantly generalize Euclidean UBGs and α-qUBGs while retaining some important geometric characteristics. Doubling UBGs provide a more flexible and robust model for pairwise distances in wireless networks in the sense that if pairwise distances in a doubling metric are perturbed, we still have a doubling metric but with a possibly different doubling dimension. The notion of doubling dimension has been introduced in [9] and it has been proposed that latencies of peer-to-peer networks and the Internet form a metric of constant doubling dimension [9, 12].

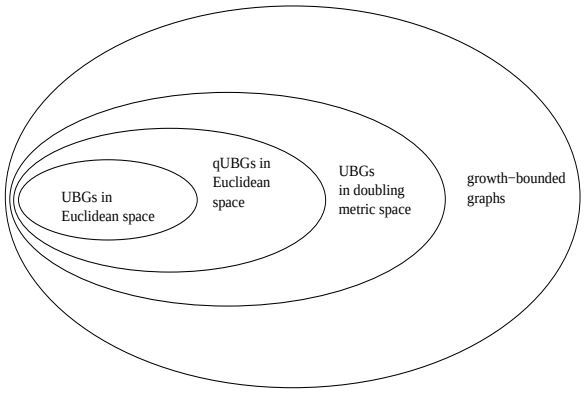

Figure 1: Shows the containment relationship of wireless network models we consider.

We provide an even faster algorithm for UBGs that reside in a Euclidean space, assuming that each node knows its coordinates in Euclidean space with respect to some globally consistent coordinate system. Our three algorithmic results are summarized below. See Figure 1 for the containment relationship of the wireless network models described thus far.

1. An $O(1)$ round algorithm that computes, for any $k\geq 2$, a k-domatic partition of size at least $(\delta_{k-1}+1)/c_k$, for some constant c_k, for UBGs that reside in Euclidean space and whose nodes are aware of their global coordinates.

2. An $O(\log^* n)$ round algorithm that computes, for any $k\geq 2$, a k–domatic partition of size at least $(\delta_{k-1}+1)/c_k$, for some constant c_k, for doubling UBGs whose nodes are able to sense distances to neighbors.

3. An $O(\log\Delta\cdot\log^* n)$ round algorithm that computes a k–domatic partition of size at least $(\delta_{k-1}+1)/c_k$, for some constant c_k, for every $k\geq 2$, for growth-bounded graphs. Here Δ is the largest degree of a vertex in the graph.

We also provide preliminary simulation results for our first algorithm, which give further insights into the quality of the domatic partitions we produce.

1.2 Model and Notation

For our algorithms we assume the synchronous communication model in which time is divided into rounds. In each round, a node can receive messages sent in the previous round, perform local computations, and broadcast a message to its neighbors. The time complexity of an algorithm is the number of rounds it needs to complete. Note that every synchronous message passing algorithm can be turned into an asynchronous algorithm with the same time complexity, but with a possibly larger message complexity.

Here we describe some additional notation that we use in the rest of the paper. For any integer $k\geq 1$ and vertex v, $N_k[v]=N_k(v)\cup\{v\}$ is the *closed k-neighborhood* of v. We will use $N[v]$ to denote $N_1[v]$. The notation $|uv|$ will be used to denote the distance between vertices u and v in

the underlying metric space. In Section 3.1 this will denote a Euclidean distance and in Section 3.2 this will denote a distance in a metric space of constant doubling dimension. For any vertex v in 2-dimensional Euclidean space and any real $\alpha > 0$ we will use $Disk(v, \alpha)$ to denote the closed disk of radius α with center v.

2. RELATED WORK

A great deal of research in wireless sensor networks has focused on the issue of minimizing power consumption. Like this paper, some of this research focuses on developing protocols for constructing "sleep-activity" schedules for nodes. In general, sleep-activity schedules tell nodes when to be active and when to go to sleep. These schedules aim to ensure that at any time there are enough active nodes to perform the tasks of the network. The premise of this line of research is that due to redundancies in node deployment, it is possible to keep the network up and running even after putting a large fraction of the nodes to sleep. This research is briefly reviewed in Section 2.1. This paper models the problem of constructing a sleep-activity schedule as the domatic partition problem. This problem has been fairly well-studied from a graph-theoretic and algorithmic point of view. This research is described in Section 2.2.

2.1 Constructing Sleep-Activity Schedules

As mentioned before, *Span* [4] is a distributed, randomized algorithm where nodes make local decisions on whether to sleep, or to join a forwarding backbone as a coordinator. Each node bases its decision on an estimate of how many of its neighbors will benefit from it being awake, and the amount of energy available to it. Simulation results have shown that with a practical energy model, system lifetime of an 802.11 network in power saving mode with Span is a factor of two better than without. However, the simulations reported in [4] are for fairly small, 2-dimensional networks of randomly distributed nodes. Much more experimental work needs to be done to understand how well the performance of Span scales and how Span performs on more realistic node distributions. Independent of whether Span is shown to be effective in simulations, a theoretical understanding of its worst case or even expected case performance would be desirable. [25] describes a protocol with goals similar to that of Span. This protocol uses a fixed size grid that allows nodes to identify which grid cell they belong to. The protocol then ensures that one node remains active from every non-empty grid cell. Our first algorithm is similar to this, but the other two algorithms proposed in this paper do not require any global positioning information and in fact run with guaranteed performance on networks that may not even be residing in Euclidean space.

In [23], the authors examine the problem of activating a small subset of sensors at any time so as to maintain coverage of a given set of targets that need to be monitored. The authors model this problem as the *set k-cover* problem defined below.

`SET k-COVER`
INSTANCE: Collection C of subsets of a set A, and a positive integer k.
QUESTION: Does C contain k disjoint covers of A, i.e., subcollections $C_1, C_2, \ldots, C_k$, where $C_i \subseteq C$, $C_i \cap C_j = \emptyset$ for all $i \neq j$, and such that every element of A belongs to at least one member of each C_i?

The set A is the set of targets that need to be covered and each member of the collection C is the set of targets that can be covered by one sensor. It is easy to see that `SET k-COVER` is a generalization of the domatic partition problem. The authors propose a greedy, centralized heuristic that seeks to minimize the coverage of sparsely covered areas within one cover with the hope that the number of disjoint sets of sensors is large. While the heuristic does not come with theoretical guarantees, the authors do show promising experimental results. Greedy heuristics typically require global coordination and it is not clear how to devise an equivalent, efficient, distributed implementation. [2] also solves the target coverage problem. This is done by reducing the set coverage problem to a maximum flow problem which is then modeled as a mixed integer program. This strategy too is a costly, centralized heuristic that does not seem amenable to distributed implementation.

[3, 19] employ `DOMATIC PARTITION` as a model of the power minimization problem. [3] models the network as a UDG and proposes a centralized heuristic that seeks to maximize the number of disjoint dominating sets. The paper does not present an analysis of the size of the domatic partition returned by the heuristic. The algorithms in [19] are distributed versions of the approximation algorithm for the domatic partition problem on general graphs described in [6]. Since the algorithm in [6] was devised for general graphs, it does not exploit any features that are specific to wireless sensor networks. The work in [3] and in [19] can be thought of as being at two ends of the spectrum in the sense that [3] assumes a rather simplistic model of a wireless network (a UDG) whereas [19] assumes a network model that is probably too general. The truth is somewhere in between and our goal in this paper is to consider network models that are neither too simplistic nor too general.

2.2 Approximation Algorithms for Domatic Partitions

`DOMATIC PARTITION` is a classical NP–complete problem [8, 6]. Since this is a maximization problem, researchers are interested in devising approximation algorithms that are guaranteed to produce domatic partitions whose size is at least some fraction f of the size of an optimal domatic partition. The approximation algorithm that seems to guarantee the largest fraction, $\frac{1}{O(\lg \Delta)}$, is due to Feige et. al [6]. In fact, the authors also show that this is the best possible approximation ration possible for general graphs unless NP $\subseteq$ DTIME($n^{O(\lg \lg n)}$).

A natural approach to solving the domatic partition problem is to greedily construct small dominating sets in an attempt to maximize the number of disjoint dominating sets. Fujita [7] has studied several greedy algorithms and shown that their performance ratio is no better than $1/\Omega(\sqrt{n})$.

Stronger results are known for some special classes of graphs. A graph G is said to be *domatically full* if its domatic number is exactly $\delta + 1$, the maximum possible. For example, the graph in Figure 2 is domatically full, but the graph in Figure 3 is not. Determining if a d–regular graph is domatically full is NP–complete for $d \geq 3$ [13, 21]. Farber [5] showed nonconstructively that *strongly chordal* graphs are domatically full. This class contains the classes of *interval graphs* and *path graphs*. Subsequently, linear time algorithms were developed [22, 20] for computing optimal domatic partitions of strongly chordal graphs. Later, Kaplan

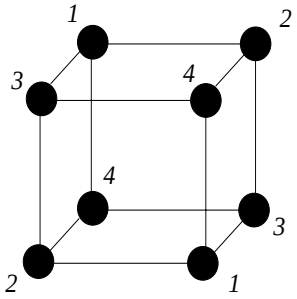

Figure 2: A 3-dimensional hypercube is domatically full. Here $\delta = 3$ and the size of the partition is 4. More generally, a d-dimensional hypercube is a d-regular graph with domatic partition of size $d+1$.

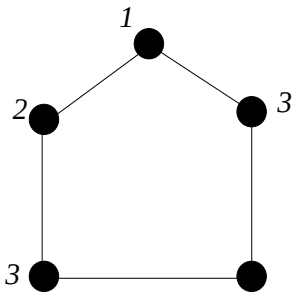

Figure 3: A graph that is not domatically full. It is not possible for a third color to be available in every closed neighborhood. Without loss of generality the vertex at the top can be colored 1 and its two neighbors can be colored as shown. Then the node at the bottom-left has to be colored 3 and that leaves the bottom-right node with either color 1 or color 2 missing from its closed neighborhood.

and Shamir [10] showed that the problem is NP–complete *split graphs* and *bipartite graphs*. The problem is shown to be NP–complete for *circular-arc graphs* [24, 1], while [18] gave a 1/4–approximation algorithm.

To our best knowledge, there is no $O(1)$–factor approximation algorithm to the domatic partition problem on UDGs. The work in this paper may be a step in that direction.

3. ALGORITHMS FOR K-DOMATIC PARTITION

In this section we describe three algorithms for finding large k-domatic partitions. For simplicity of exposition, we describe our algorithms for UDGs first and later point out why the algorithms work in more general settings. We start with a simple constant-time algorithm that assumes that all nodes know their positions in a global coordinate system. This algorithm works for UBGs that reside in Euclidean space. Our second algorithm drops knowledge of global coordinates and instead assumes that pairwise distances between neighboring nodes are known. In particular, we assume that each node $u \in V(G)$ knows $|uv|$ for all $v \in N(u)$. This algorithm runs in $O(\log^* n)$ time. We do not in any way rely on the fact that our graph resides in 2-dimensional Euclidean space and it will be clear that our algorithm works for UBGs in a metric space with constant doubling dimension. Our third algorithm drops all reliance on geometric information, using connectivity information only. This algorithm works for bounded growth graphs and runs in $O(\log \Delta \cdot \log^* n)$ time. All our algorithms are deterministic.

3.1 Using a global coordinate system

Suppose that the input graph is a UDG G and assume that each node knows its position in a global coordinate system. Our algorithm can be described informally as follows. Place on the plane, an infinite grid of square cells of dimensions $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$. This induces a partition $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ of $V(G)$ such that each block V_i corresponds to a non-empty cell and contains all the vertices in that cell. Note that due to the dimensions of each cell, each V_i is a clique. Consider each block V_i and assign a distinct color $r \in \{1, 2, \ldots, |V_i|\}$ to each vertex in V_i. In Theorem 1 we will show that for many colors r, the set of all vertices colored r form a k-dominating set. Below, we describe the algorithm in more detail.

Algorithm: k-DP1

1. Let the coordinates of vertex v be (x_v, y_v). Each node v determines the ordered pair (i, j) of integers such that
$$\frac{i}{\sqrt{2}} \leq x_v < \frac{i+1}{\sqrt{2}} \text{ and } \frac{j}{\sqrt{2}} \leq y_v < \frac{j+1}{\sqrt{2}}.$$
2. Denote the 4-tuple (x_v, y_v, i, j) by ID_v. Each node v broadcasts ID_v to all neighbors.
3. Each node v receives ID_u from each neighbor $u \in N(v)$.
4. Each node v constructs the set $S_v = \{(x_v, y_v)\} \cup \{(x_u, y_u) \mid u \in N(v) \text{ and } ID_u = (x_u, y_u, i, j)\}$.
5. Each node v sorts S_v in lexicographic order and assigns to itself the color r, where r is the rank of (x_v, y_v) in the sorted list S_v.

3.1.1 Correctness

In Step (1), each node v determines the south-west corner of the square cell that it belongs to. Then, in Steps (2)-(3), nodes exchange with neighbors, their coordinates and identities of the cells they belong to. For each node v, this information is bundled into the 4-tuple ID_v. In Step (4), each node v gathers into a set S_v the coordinates of all neighbors that lie in the same cell as it does. Note that for any two vertices u and u' that lie in a cell, $S_u = S_{u'}$. Therefore, the set $\{S_v \mid v \in V(G)\}$ is the clique partition $\mathcal{V}$ induced by the cells. Finally, in Step (5), each node v in each block V_i, assigns itself a distinct color $r \in \{1, 2, \ldots, |V_i|\}$.

Algorithm k-DP1 consists of one local broadcast by each node and therefore runs in a constant number of communication rounds. Let D_r be the set of vertices colored r by k-DP1. We will now investigate the quality of the partition $\{D_r \mid r = 1, 2, \ldots\}$. Let c_k denote the maximum number of $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$ square cells that can intersect a disk of radius k. It is easy to see that $c_k = \Theta(k^2)$ and specifically $c_1 = 16$.

THEOREM 1. *For any r, $1 \leq r \leq (\delta_{k-1}+1)/c_{k-1}$, the set D_r computed by k-DP1 is a k-dominating set of G.*

PROOF. To obtain a contradiction, suppose that for some $v \in V(G)$, D_r does not k-dominate v, that is, $D_r \cap N_k[v] = \emptyset$. This, of course means that $D_r \cap N_{k-1}[v] = \emptyset$ as well. Every vertex $u \in N_{k-1}[v]$ lies in a disk of radius $k-1$ centered at v.

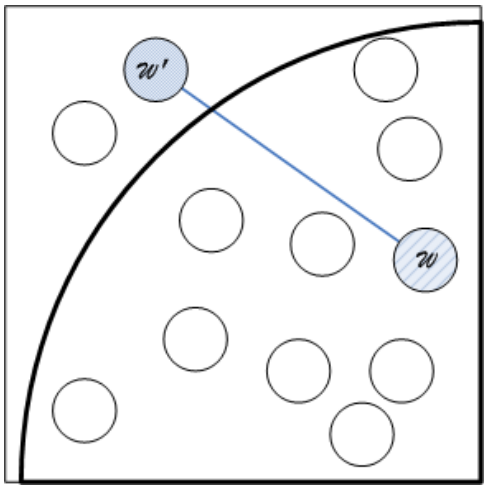

Figure 4: An illustration of the proof of Theorem 1. The figure shows a portion of the boundary of the disk centered at v of radius $k-1$ and it shows a grid cell intersecting this boundary. w is a node with color $\geq r$ that lies in the intersection of the grid cell and the disk. Thus $w \in N_{k-1}[v]$. The proof of Theorem 1 shows that there is at least one vertex w' that is colored r and lies in the grid cell. Since each grid cell is sufficiently small and induces a clique, $d(w,w') \leq 1$, implying that $w' \in N_k[v]$.

Since at most c_{k-1} grid cells intersect this disk, at most c_{k-1} blocks of the partition $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ induced by the grid cells, intersect $N_{k-1}[v]$. Thus, at most c_{k-1} vertices in $N_{k-1}[v]$ are in D_j, for any $j < r$. Thus at most $c_{k-1} \cdot (r-1)$ vertices in $N_{k-1}[v]$ are in $D_1 \cup D_2 \cup \cdots \cup D_{r-1}$. Since $r \leq (\delta_{k-1}+1)/c_{k-1}$, we have that $c_{k-1} \cdot (r-1) < \delta_{k-1}+1$. Thus there is at least one vertex $w \in N_{k-1}[v]$ that is not in $D_1 \cup D_2 \cup \cdots \cup D_{r-1}$.

Suppose that $w \in V_\ell$. Since w has a color $\geq r$, there must be a $w' \in V_\ell$ colored r. Since V_ℓ is a clique, $d(w,w') \leq 1$. This along with the fact that $w \in N_{k-1}[v]$, implies that $w' \in N_k[v]$, thereby contradicting the fact that D_r does not k-dominate v. See Figure 4 for an illustration of this proof. □

The proof of Theorem 1 implies something stronger than what is stated in Theorem 1. The proof tells us not only about the size of a k-domatic partition, it tells us about the sizes of a $(k-1)$-domatic partition, a $(k-2)$-domatic partition, etc., all the way down to the size of a 2-domatic partition. For example, suppose that $k = 4$. Then the proof tells us that the colors $1, 2, \ldots, (\delta_3+1)/c_3$ 4-dominating, but in addition it tells us that the colors $1, 2, \ldots, (\delta_1+1)/c_1$ are 2-dominating and the colors $1, 2, \ldots, (\delta_2+1)/c_2$ are 3-dominating. This observation leads to the following corollary.

COROLLARY 1. *Let $k \geq 2$. For any $j \geq 1$, let $U_j = (\delta_j+1)/c_j$. Then, for any k, $1 \leq j \leq k$, and any r, $1 \leq r \leq \max\{U_1, U_2, \ldots, U_{j-1}\}$, D_r is j-dominating.*

All of the discussion in this subsection assumed that the given network resides in 2-dimensional Euclidean space. All of these ideas extend in a straightforward manner to UBGs in d-dimensional Euclidean space. In this case, the algorithm starts by placing an infinite grid of d-dimensional hypercubes of dimensions $\frac{1}{\sqrt{d}} \times \frac{1}{\sqrt{d}} \times \cdots \frac{1}{\sqrt{d}}$. Each hypercube has diameter 1 and therefore each non-empty hypercube induces a clique in G. The rest of the algorithm and proof proceeds as in the 2-dimensional case.

3.1.2 Running Time and Message Size

In Step (2) of k-DP1, each node v broadcasts ID_v to all neighbors. There is no other communication in this algorithm. The rest of the steps involve local computations. This implies that the algorithm runs in $O(1)$ communication rounds. During Step (2), each node v broadcasts a 4-tuple (x_v, y_v, i, j) to neighbors. The number of bits needed for i and j is proportional to the number of bits used for x_v and y_v. Therefore, the size of v's message is proportional to the number of bits used to store its coordinates. In general, the size of messages in k-DP1 is a constant times the maximum number of bits used to store the coordinates of a node in the network. In summary we have the following result.

THEOREM 2. *For any $k \geq 2$ and fixed $d \geq 2$, Assuming that nodes know their locations in a global coordinate system, Algorithm k-DP1 computes a k-domatic partition of size at least $(\delta_{k-1}+1)/c_{k-1}$ for a given UBG in d-dimensional Euclidean space in $O(1)$ rounds using messages of size $O(K)$, where K is the maximum number of bits used to represent the coordinates of a node.*

3.1.3 Discussion of Theorem 1

Theorem 1 guarantees that for any fixed k, the size of the k-domatic partition computed by k-DP1 is within a constant factor of the optimal $(k-1)$-domatic partition. For example, since $c_1 = 16$, 2-DP1 computes a 2-domatic partition of size at least $(\delta_1+1)/16$. In other words, 2-DP1 computes a 2-domatic partition whose size is at least 1/16th the size of a largest possible 1-domatic partition. Experiments show that this estimate is quite conservative and in general, our algorithm returns 2-domatic partitions that are more than half the size of a largest possible 1-domatic partition.

In Tables 1 and 2, we show results obtained by running 2-DP1 and 3-DP1 on randomly generated UDGs. Table 1 contains results obtained by running 2-DP1 and 3-DP1 on UDGs that were generated by distributing points uniformly at random while Table 2 contains results obtained by running the same algorithms on UDGs that were generated in a way so that points clustered in some regions and were distributed is sparsely in other regions. The first row in the tables shows n, the size of the network, the next three rows show corresponding values of δ_1, δ_2, and δ_3, and the next two rows show the sizes of the 2-domatic partition and the 3-domatic partition returned by running 2-DP1 and 3-DP1 respectively. Comparing the second row (δ_1) with the fifth row (size of 2-domatic partition), we see that for all values of n, the size of the 2-domatic partition is at least half the value of δ_1. Comparing the third row (δ_2) with the sixth row (size of the 3-domatic partition), we see that the size of the computed 3-domatic partition is about a third of δ_2. Theorem 1 only guarantees that the size of the computed 3-domatic partition is at least $1/c_2$ times δ_2, where $c_2 = 32$. Again we see that Theorem 1 is quite conservative in its predictions. Our results also reveal that a fraction of these k–dominating sets are connected. The number of connected 2-dominating sets

n	50	60	100	125	150
δ_1	7	9.2	15.5	19.2	22.7
δ_2	15.9	18.1	32.7	42.1	52.8
δ_3	23.4	29.3	50.9	64.3	80.1
size ($k = 2$)	5.4	6.4	10.5	12.6	16.6
size ($k = 3$)	6.1	7.6	12.5	14.2	17.6
connected ($k = 2$)	1.7	2	3.5	4.5	5.6
connected ($k = 3$)	1.5	1.7	3.8	5.7	4.5

n	175	200	225	250	275
δ_1	30.1	30.8	35.2	40.6	41.6
δ_2	62.1	71.0	78.5	86.4	97
δ_3	95.2	107.2	122.7	137	151.2
size ($k = 2$)	18.3	20.6	22.5	26.6	27.6
size ($k = 3$)	20.4	23.5	25	28.1	30.1
connected ($k = 2$)	5.9	7.3	7.8	9.7	9.8
connected ($k = 3$)	7.9	8.3	9.2	11.2	10.8

Table 1: Table showing results obtained by running 2-DP1 and 3-DP1 on UDGs generated uniformly at random.

n	125	130	135	140	205
δ_1	5.4	9.2	13.3	15.0	20.5
δ_2	31.2	33.6	35.4	36.6	56.7
δ_3	61.5	63.7	64.8	66.9	101.6
size ($k = 2$)	8.3	9.1	9.5	9.7	14.4
size ($k = 3$)	10.9	11.9	11.2	11.3	16.2
connected ($k = 2$)	3.7	3.8	2.9	3.5	4.9
connected ($k = 3$)	4.1	4.4	3.8	3.5	4.7

Table 2: Table showing results obtained by running 2-DP1 and 3-DP1 on clustered UDGs generated randomly.

and the number of connected 3-dominating sets are reported in the last two rows, respectively. By comparing the fifth row (size of 2-domatic partition) with the seventh row (number of connected 2-dominating sets) in both tables, we see that about a third of the 2-dominating sets are connected. Obtaining connectivity was not a goal of our algorithms and so in a sense we get this property for free simply because of the density of the node deployment.

Let D be a 2-dominating set of the network that is a member of the 2-domatic partition returned by Algorithm 2-DP1. As mentioned in the introduction, a node v may not have any node from D in its transmission range and so one way for v to reach a node in D is to increase its transmission range. From the proof of Theorem 1, it can be observed that there is always some node in D that lies in one of the grid cells that intersects $Disk(v, 1)$. Thus, in order to reach D, v just has to increase its transmission range to completely cover all of the grid cells that intersect $Disk(v, 1)$. For any fixed $\epsilon > 0$, we can make the grid cells have dimensions $\epsilon \times \epsilon$, thereby requiring v to increase its transmission range by some function of ϵ. Thus, the 2-dominating set D computed by 2-DP1 can be viewed as an arbitrarily good approximation of a 1-dominating set.

3.2 Using pairwise distances only

In the previous section, we assumed that nodes of the input network reside in d-dimensional Euclidean space for some fixed d *and* that these nodes are aware of their d-dimensional coordinates with respect to some fixed global coordinate system. Here we assume that nodes are not aware of coordinates, but can sense distances to neighbors. In fact, for the algorithms described in this section, it is not even necessary for the nodes to reside in Euclidean space. In other words, it is not necessary for the pairwise distances between nodes to form a Euclidean metric; it is sufficient if these distances induce a metric with constant doubling dimension. For simplicity of exposition, our initial discussion assumes that the given network G is a UDG. We also assume that nodes in the network have unique IDs; the ID of node v is denoted by ID_v.

Recall that in Algorithm k-DP1 places a grid of small enough square cells on the plane and constructs a clique-partition $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ of $V(G)$. Given a partition $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ of $V(G)$ and an arbitrary subset $S \subseteq V(G)$, the *density* of S with respect to $\mathcal{V}$ is the size of the set $\{i \mid V_i \cap S \neq \emptyset\}$. In other words, the density of S with respect to $\mathcal{V}$ is the number of blocks in $\mathcal{V}$ that intersect S. The key property of the clique-partition $\mathcal{V}$ computed by k-DP1 is this:

> **Bounded Density Property.**
> For each integer $j \geq 1$, there is a constant c_j such that the density of each neighborhood $N_j[u]$ with respect to $\mathcal{V}$ is bounded above by c_j.

Given just the pairwise distances how do we compute a clique-partition with the bounded density property? We first describe the algorithm informally. Let $G_{1/2}$ be the spanning subgraph of G with edge set $\{\{u, v\} \mid |uv| \leq 1/2\}$. Since each node u knows the distances $|uv|$ to all neighbors $v \in N(u)$, node u can identify who its neighbors in $G_{1/2}$ are. First a maximal independent set (MIS) I in $G_{1/2}$ is computed. Then each node $u \in V(G) - I$ "attaches" itself to a node $v \in I$ that is its neighbor in $G_{1/2}$. Since I is maximal such a node v is guaranteed to exist. Each block in the partition we seek, consists of a vertex $v \in I$ along with all the nodes in $V(G) - I$ that have attached themselves to v. Note that since the distance between v and a vertex that attaches itself to v is at most $1/2$, the distance between any pair of vertices in a block is at most 1 and therefore each block induces a clique in G. Later we show that this clique-partition has the bounded density property.

Algorithm: k-DP2

1. Let $G_{1/2} = (V(G), E_{1/2})$, where $E_{1/2} = \{\{u, v\} \mid |uv| \leq 1/2\}$. Compute an MIS I of $G_{1/2}$.
2. Each node $v \in I$ broadcasts ID_v to its neighbors.
3. Each node $u \in V(G) - I$ receives at least one ID from a node at distance at most 1/2 from u. From among these, u picks one ID. Denote by L_u, the ID picked by u. Node u broadcasts the ordered pair (ID_u, L_u) to its neighbors.
4. Each node $v \in I$ receives ordered pairs of IDs from neighbors and constructs the set:

   $S_v = \{ID_v\} \cup \{id \mid (id, ID_v)$ is received from neighbor$\}$.

5. Each node $u \in V(G) - I$ receives ordered pairs of IDs from neighbors and constructs the set:

   $$S_u = \{ID_u\} \cup \{L_u\} \cup \{id \mid (id, L_u) \text{ is received from a neighbor}\}.$$

6. Each node $v \in V(G)$ colors itself r, where r is the rank of ID_v in the sorted set S_v. Let this coloring be denoted by χ. Note that this coloring is not necessarily proper.

3.2.1 Correctness

In Step (1), an MIS I of $G_{1/2}$ is computed. We assume that whenever a node u receives a message from node v, it can sense the distance $|uv|$ between itself and v. Therefore, to construct $G_{1/2}$, each node starts by broadcasting a message. Following this, each node u can identify the subset of neighbors in G that are at distance $\leq 1/2$ from it. In Step (2), the vertices in the MIS I, announce their IDs to all neighbors. Since I is an MIS in $G_{1/2}$, in Step (3) of the algorithm node $u \in V(G) - I$ receives at least one ID from a neighbor at distance $\leq 1/2$ from it. Node u picks one such ID. We denote u's choice of an ID by L_u to indicate that this amounts to choosing a leader to attach to. For any $v \in I$, we say that v's *group* consists of v along with all neighbors u of v in $G_{1/2}$ for which $L_u = ID_v$. More precisely, let $group(v) = \{v\} \cup \{u \mid L_u = ID_v\}$. Note that for all $v \in I$, $group(v)$ induces a clique in G since for any two vertices $x, y \in group(v)$ satisfy $|xv| \leq 1/2$, $|yv| \leq 1/2$, and therefore $|xy| \leq 1$. When u broadcasts (ID_u, L_u) in Step (3), it is announcing its intention to join L_u's group. After hearing this announcement from nodes $u \in V(G) - I$, in Step (4), each node $v \in I$ constructs a set S_v containing the IDs of the nodes in $group(v)$. Similarly, in Step (5), each node $u \in V(G) - I$ constructs a set S_u containing the IDs of nodes that belong to the same group as it does. This situation after Step (5) can be summarized thus.

LEMMA 1. *For each node $v \in I$, the subgraph induced by $group(v)$ is a clique. Each node $v \in I$ constructs a set $S_v = \{ID_u \mid u \in group(v)\}$. For each node $u \in group(v)$, $S_u = S_v$.*

After Step (5), for each $v \in I$, members of $group(v)$ know the identity of the group and the only thing left to do is a coloring so that members in $group(v)$ are assigned distinct colors from the set $\{1, 2, \ldots, |S_v|\}$. This happens in Step (6).

As before, for each $r = 1, 2, \ldots$ let $D_r = \{v \in V(G) \mid \chi(v) = r\}$. To obtain the same result as in Theorem 1 for Algorithm k-DP2, we need to show that the partition $\{group(v) \mid v \in I\}$ has the bounded density property.

LEMMA 2. *For each integer $j \geq 1$, there is a constant c_j such that the density of each neighborhood $N_j[u]$ with respect to $\{group(v) \mid v \in I\}$ is bounded above by c_j.*

PROOF. Fix an integer $j \geq 1$ and a vertex $u \in V(G)$. All vertices in $N_j[u]$ lie in $Disk(u, j)$. Therefore the density of $N_j[u]$ with respect to $\{group(v) \mid v \in I\}$ is bounded above by number of vertices in I that lie in $Disk(u, j+1/2)$. Since any two vertices in I are separated by a distance $> 1/2$, disks of radius 1/4 centered at vertices in I are pairwise disjoint. Therefore, if a vertex $v \in I$ lies in $Disk(u, j + 1/2)$ then $Disk(v, 1/4)$ is completely contained in $Disk(v, j + 3/4)$. The total number of disjoint disks of radius 1/4 completely contained in $Disk(u, j+3/4)$ is bounded above by $16(j+1)^2$. Therefore, there is a constant $c_j \leq 16(j + 1)^2$ such that the density of $N_j[u]$ with respect to $\{group(v) \mid v \in I\}$ is bounded above by c_j. □

Algorithm k-DP2 is well defined for UBGs in which nodes can sense distances to neighbors. For Lemma 1, it is required that the pairwise distances form a metric, since we use the fact that two neighbors of a node u at distance at most 1/2 from u, are at distance at most 1 from each other. The proof of Lemma 2 relies on the fact that the number of disjoint disks of radius 1/4 that can be packed into a disk of radius $j + 3/4$ is bounded above by a quantity that depends only on j. This fact is true not just for 2-dimensional Euclidean space, but is in general true for any metric space of constant doubling dimension. Therefore, the two lemmas above and the proof of Theorem 1 gives us the following result.

THEOREM 3. *Let G be a doubling UBG. For any r, $1 \leq r \leq (\delta_{k-1} + 1)/c_{k-1}$, the set D_r computed by k-DP2 is a k-dominating set of G.*

3.2.2 Running time and Message Size

Since G is a doubling UBG, we see that $G_{1/2}$ is also a doubling UBG by scaling distances in $G_{1/2}$ by a factor of 2. Therefore, using the deterministic MIS algorithm in [15] we can implement Step (1) of Algorithm k-DP2 in $O(\log^* n)$ rounds. The rest of the steps in the algorithm involve three broadcasts and some local computations. Thus the entire algorithm runs in $O(\log^* n)$ rounds.

The MIS algorithm presented in [15] uses messages of size $O((\log^* n + \Delta) \cdot \log n)$. Thus Step (1) of Algorithm k-DP2 uses messages of size $O((\log^* n + \Delta) \cdot \log n)$. Step (2) involves broadcasting one ID and Step (3) involves broadcasting 2 IDs. Therefore the message size in Steps (2)-(3) is $O(\log n)$. The last three steps involve no communication; only local computations. Thus the overall message size in Algorithm k-DP2 is $O((\log^* n + \Delta) \cdot \log n)$. In the worst case, this can be $\Theta(n \log n)$. It would be nice to be able to reduce this to $O(\log n)$ and we believe it is worth reexamining the protocol in [15] to see if this can be done. Our final result is given below.

THEOREM 4. *For any $k \geq 2$, Algorithm k-DP2 computes a k-domatic partition of size at least $(\delta_{k-1} + 1)/c_{k-1}$ for a given doubling UBG in $O(\log^* n)$ rounds, using messages of size $O((\log^* n + \Delta) \cdot \log n)$.*

3.3 Using connectivity information only

Algorithm k-DP2 relied critically on the ability of nodes to sense distances. It also relied on the fact that these distances

formed a metric of constant doubling dimension. We now generalize our results further by assuming that no distance information is available. We first describe how to compute a large 2-domatic partition and later point out the extension to k-domatic partitions. As before, for simplicity of exposition our initial discussion assumes that the input network G is a UDG. We assume that nodes have unique IDs; node v's ID will be denoted ID_v.

Using only connectivity information, it is not clear how to quickly compute a clique-partition that has the bounded density property. One possible approach (that does not work) might be to compute an MIS I of G and then have each node $u \in V - I$ attach itself to an arbitrary neighbor in I. This induces a partition of V into subsets of diameter at most 2. This partition does have the bounded density property because G is assumed to be a UDG and therefore the set I restricted to any closed neighborhood $N_j[u]$ has bounded size (that depends only on j). In fact, this partition would have the bounded density property even if G were a growth-bounded graph. However, this partition is not a clique-partition and it is not clear how to convert it into one.

So we modify our approach and compute a partition that we call a *uniform partition*. A partition $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ of $V(G)$ is called a *uniform partition* if the following two conditions are satisfied:

(i) Each V_i induces a subgraph of G of diameter ≤ 2.

(ii) There is a constant C such that for each i, $1 \leq i \leq t$, $|V_i| \geq (\delta_1 + 1)/C$.

The following lemma shows that it is a small step from a uniform partition to a 2-domatic partition.

LEMMA 3. *Let $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ be a uniform partition of G. For each i, $1 \leq i \leq t$, arbitrarily color the vertices in V_i with distinct colors chosen from $\{1, 2, \ldots, |V_i|\}$. For each integer $r \geq 1$, let D_r be the set of vertices colored r. Then, for each integer r, $1 \leq r \leq \lceil(\delta_1 + 1)/C\rceil$, D_r is a 2-dominating set of G.*

PROOF. Consider an arbitrary vertex $v \in V(G)$ and an arbitrary color r, $1 \leq r \leq \lceil(\delta_1 + 1)/C\rceil$. Suppose that v belongs to block V_i. Since V_i is large enough, that is, $|V_i| \geq (\delta_1 + 1)/C$, there is a vertex in V_i colored r. Since the diameter of $G[V_i]$ is at most 2, there is a vertex colored r at distance at most 2 from v. Hence, D_r is a 2-dominating set. □

We now informally describe an algorithm to quickly compute a uniform partition. Suppose that I is an MIS in G. One way to get a partition of $V(G)$ into components of diameter at most 2 is to simply allow each vertex $u \in V(G) - I$ to attach itself to some neighbor that belongs to I. However, blocks in the resulting partition need not be large enough. To guarantee a lower bound on the size of each block, we first form the graph $H = (I, E_H)$, where $E_H = \{\{v, v'\} \mid v, v' \in I$ and v and v' have a common neighbor$\}$. Thus a pair of vertices in I are connected in H if the hop-distance between them in G is at most 2. Note that since G is a UDG, for any $v \in I$, the number of vertices $v' \in I \cap N_2[v]$ is bounded above by a constant. Therefore, $\Delta(H)$ is bounded above by a constant, say L. We then compute an $(L+1)$-coloring of this graph using colors from $\{1, 2, \ldots, L+1\}$. Then for each color $r = 1, 2, \ldots, L+1$, all of the vertices in I of color r try to acquire "followers." Each vertex $v \in I$ of color r, has a budget of how many followers it can acquire, thereby leaving enough followers available for vertices in I of color larger than r. Specifically, each vertex $v \in I$ of color r acquires $\lfloor \delta_1/(L+1) \rfloor$ followers during its turn. Only the vertices in I of the same color will try to acquire followers at the same time. Since a pair of vertices in I of the same color do not have a common neighbor, there is no contention for followers. Furthermore, by imposing a constraint on the number of followers that a node in I can acquire in each round, we ensure that there are enough followers for all vertices in I. We now describe this algorithm (called 2-DP3) in detail.

Algorithm: 2-DP3

1. Compute an MIS I of G.
2. Let G^2 denote the square of the graph G. So G^2 has vertex set $V(G)$ and edge set $E^2 = \{\{u, v\} \mid u, v \in V(G)$ and $d(u, v) \leq 2\}$. Let $H = G^2[I]$. This is the subgraph of G^2 induced by I. Let $L = \Delta(H)$. Compute a proper $(L+1)$-vertex coloring of H. Denote this coloring by χ.
3. Each node $u \in V(G) - I$ sets a variable $status_u \leftarrow$ *available*. Each node $v \in I$ sets a variable $status_v \leftarrow$ *unavailable*.
4. For each color $r = 1, 2, \ldots, L+1$ used by χ, repeat the following steps.
   (a) Each node $u \in V(G) - I$ whose status is available, broadcasts its ID, ID_u, to neighbors.
   (b) Each node $v \in I$ colored r by χ receives an ID from each available neighbor and constructs the set $C_v = \{ID_u \mid u \in N(v)$ and $status_u = available\}$.
   (c) Each node $v \in I$ colored r by χ picks the smallest $\lfloor \delta_1/(L+1) \rfloor$ IDs from C_v and places these in S_v. Node v then broadcasts $\{ID_v\} \cup S_v$ to neighbors. For this step, it is not necessary that node v know δ_1. It is sufficient for v to instead use the smallest vertex degree in its neighborhood instead of δ_1.
   (d) Each node $u \in V(G) - I$, whose status is available, may receive a set S of IDs from a neighbor in I. Node u then checks if $ID_u \in S$ and if so u sets $status_u \leftarrow$ *unavailable* and $S_u \leftarrow S$.
5. Each unavailable node v computes the rank r of ID_v in S_v and colors itself r. Each available node colors itself 1. Let this coloring of vertices be denoted χ'. Note that this vertex coloring need not be proper.

3.3.1 Correctness

In Step (1), an MIS I of G is computed. In Step (2), the subgraph H of G^2 induced by I is formed and a proper $(\Delta(H)+1)$-vertex coloring χ of H is computed. It is assumed that these colors come from the palette $\{1, 2, \ldots, \Delta(H)+1\}$. Thus each vertex $v \in I$ has two associated numbers, ID_v and $\chi(v)$. The following lemma shows that the number of colors used by χ is bounded above by a constant.

LEMMA 4. *Let $H = G^2[I]$. Then $\Delta(H) \leq 25$.*

PROOF. If $v, v' \in I$ are neighbors in H then $|vv'| \leq 2$. Therefore, for any $v \in I$, all neighbors of v in H are in $Disk(v, 2)$. Since I is an independent set in G, $|vv'| > 1$ for any pair of vertices $v, v' \in I$. Therefore, the disks in $J = \{Disk(v', 1/2) \mid v'$ is a neighbor of v in $H\}$ are pair-

wise disjoint and are all contained in $Disk(v, 5/2)$. Therefore, $|J| < \frac{\pi(5/2)^2}{\pi(1/2)^2} = 25$. □

After Step (2), each node $u \in V(G) - I$ is available for attaching itself to a neighbor $v \in I$. This is indicated in Step (3) by setting $status_u$ to *available* for all nodes $u \in V(G) - I$. For notational convenience, nodes in I also have a *status* variable and that is initialized to *unavailable*. The coloring χ of I is used to schedule the nodes in I in Step (4). In particular, for each $r = 1, 2, \ldots, L+1$, considered in that order, all nodes in I colored r execute Steps 4(b)-(c) in parallel. In Step 4(a) all available nodes announce their availability to neighbors by broadcasting their IDs. Note that since I is maximal, this announcement by a node u is heard by at least one node in I. In Step 4(b) each node $v \in I$ colored r receives IDs from neighboring available nodes and places these in a set C_v. Thus C_v is the set of IDs of nodes that are available and neighboring v. In Step 4(c), each node $v \in I$ colored r picks $\lfloor \delta_1/(L+1) \rfloor$ IDs from C_v. For this step to be well-defined, it must be the case that $|C_v| \geq \lfloor \delta_1/(L+1) \rfloor$, as shown in the following lemma.

LEMMA 5. *For any node $v \in I$, the set C_v constructed in Step 4(b) has size $\geq \lfloor \delta_1/(L+1) \rfloor$.*

PROOF. Suppose that the degree of v in H is α. The total number of neighbors of v, available in Step 5(a) is at least $degree(v) - \alpha \cdot \lfloor \delta_1/(L+1) \rfloor$. This is because if a neighbor u of v is unavailable, it is because u has attached itself to some $v' \in I$, $v' \neq v$. This means that v' is 2 hops away from v and is therefore a neighbor of v in H. The maximum number of vertices u that can be attached to v' is $\lfloor \delta_1/(L+1) \rfloor$ and the maximum number of candidates for v' is α. Since $degree(v) \geq \delta_1$ and $\alpha \leq L$, we get that at least

$$\delta_1 - L \cdot \lfloor \frac{\delta_1}{L+1} \rfloor \geq \frac{\delta_1}{L+1}$$

neighbors of v are available. □

After $\lfloor \delta_1/(L+1) \rfloor$ colors in C_v are picked by v, these are placed in a set S_v and the set $\{ID_v\} \cup S_v$ is broadcast by v to all neighbors. In Step 4(d) all neighbors of v, whose colors are in S_v, on receiving notification of their membership in S_v, set their status to unavailable and take note of $\{ID_v\} \cup S_v$. The following lemma summarizes the situation immediately after Step (4).

LEMMA 6. *For any $v \in I$, define $group(v) = \{v\} \cup \{u \in N(v) \mid ID_u \in S_v\}$. Then for all $v \in I$, (i) $group(v)$ induces a subgraph of G of diameter ≤ 2, (ii) $|group(v)| \geq (\delta_1 + 1)/(L+1)$, and (iii) for any $v, v' \in I$, $group(v)$ and $group(v')$ are disjoint if $v \neq v'$.*

PROOF. Let v be an arbitrary vertex in I. Every vertex in $group(v) - \{v\}$ is a neighbor of v and therefore $group(v)$ induces a subgraph of G of diameter ≤ 2. Since $|S_v| = \lfloor \delta_1/(L+1) \rfloor$,

$$|group(v)| \geq 1 + \lfloor \delta_1/(L+1) \rfloor \geq \frac{\delta_1 + 1}{L+1}.$$

To prove property (iii), we consider two separate cases depending on whether v and v' have been assigned the same color by χ or not. Suppose $\chi(v) \neq \chi(v')$. Without loss of generality, assume that $\chi(v) < \chi(v')$. Then v proceeds first and all the nodes that join $group(v)$ set their status to unavailable by the time v' is processed. Therefore, none of these nodes will join $group(v')$. Now suppose that $\chi(v) = \chi(v')$. In this case, v and v' are processed simultaneously. However, since v and v' have the same color in H, they are separated from each other by at least 3 hops in G. Hence, v and v' have no common neighbors and therefore $group(v)$ and $group(v')$ are disjoint. □

Note that even though $\{group(v) \mid v \in I\}$ is a collection of disjoint subsets of vertices, it is not necessarily a partition of G because after Step (4) there may still be some vertices $u \in V(G) - I$ with $status_u = available$. However, we still get a large enough 2-domatic partition, as the following lemma shows. Therefore, in Step (5) each available vertex (arbitrarily) colors itself 1.

LEMMA 7. *For any integer r, $1 \leq r \leq \lceil (\delta_1 + 1)/(L+1) \rceil$, let D_r be the set of vertices in $V(G)$ that are colored r by χ' (in Step (5) of* 2-DP3*). Then D_r is a 2-dominating set of G.*

PROOF. First consider an arbitrary vertex $v \in I$. Since, $|group(v)| \geq (\delta_1 + 1)/(L+1)$, since every vertex in $group(v)$ gets a distinct color in $\{1, 2, \ldots, |group(v)|\}$, and since every vertex in $group(v)$ is at a distance at most 1 from v, it follows that for any r, $1 \leq r \leq \lceil (\delta_1 + 1)/(L+1) \rceil$, v is dominated (and hence 2-dominated) by D_r. Now consider a vertex $u \in V(G) - I$. Since u has a neighbor $v \in I$, using the same argument as above with respect to $group(v)$, we obtain that for any r, $1 \leq r \leq \lceil (\delta_1 + 1)/(L+1) \rceil$, u is 2-dominated by D_r. □

The algorithm 2-DP3, as described, is well-defined for arbitrary graphs. In the proof of correctness of the algorithm, Lemma 4 used the fact that G is a UDG. However, the same result (but with a possibly different constant) holds even if G is not a UDG, but is a bounded growth graph. In fact, the definition of a bounded growth graph tells us that for any vertex $v \in V(G)$, the number of vertices $v' \in I$ that are in the 2-neighborhood of v is bounded above by the constant $f(2)$. The next two lemmas in the proof of correctness of 2-DP3 follow from Lemma 4 and therefore hold for bounded growth graphs as well. We get the following theorem.

THEOREM 5. *For any given growth bounded graph G, the algorithm* 2-DP3 *computes a 2-domatic partition of size at least $\lceil (\delta_1 + 1)/C \rceil$ for some constant C.*

The extension of 2-DP3 to k-domatic partitions is straightforward. Define a *k-uniform partition* of G as a partition $\mathcal{V} = \{V_1, V_2, \ldots, V_t\}$ of $V(G)$ such that for each i, $1 \leq i \leq t$, the diameter of $G[V_i]$ is at most k and $|V_i| \geq (\delta_{k-1} + 1)/c_k$ for some constant c_k. Going from a uniform partition to a 2-domatic partition is a small step, as described in Lemma 3. As in that lemma, coloring the vertices in each V_i with distinct colors from $\{1, 2, \ldots, |V_i|\}$ yields a k-domatic partition of size at least $\lceil (\delta_{k-1} + 1)/c_k \rceil$. Constructing a k-uniform partition can be done in a manner very similar to what is described in 2-DP3. This discussion yields the following theorem.

THEOREM 6. *For any given growth bounded graph G and for any fixed integer $k \geq 2$, there is a distributed algorithm that computes a k-domatic partition of size at least $\lceil (\delta_{k-1} + 1)/c_k \rceil$ for some constant c_k.*

3.3.2 Running time and Message size

In Step (1) an MIS of a growth-bounded graph is computed. Using the algorithm in [14] we can do this is $O(\log \Delta \cdot \log^* n)$ time. Step (2) computes a proper $(\Delta+1)$-coloring of a graph with bounded degree. This can be done in $O(\log^* n)$ rounds using Linial's algorithm [16]. In Step (3), nodes perform some local computations. Step (4) is repeated $O(1)$ times (since L is a constant) and in each repetition consists of a broadcast-receive-broadcast-receive sequence. Step (5) involves some local computations. Thus, Steps (3)-(5) take $O(1)$ rounds. The overall running time of the algorithm (dominated by the running time of Step (1)) is $O(\log \Delta \cdot \log^* n)$.

The message size of Step (1) is $O(\log n)$, because that is the message size used in the algorithm in [14]. The message size of Step (2) is also $O(\log n)$ since the maximum degree of H is bounded. Step (3) is a local computation and involves no communication. Since unique IDs in an n-node vertex can be represented using $O(\log n)$ bits, Step 4(a) has message size of $O(\log n)$. Step 4(b) is a local computation and involves no communication. Step 4(c) uses messages of size $O(\Delta \cdot \log n)$ and this dominates the message size of the algorithm. Steps 4(d) and (6) are local computations. Thus the overall message complexity of the algorithm is $O(\Delta \cdot \log n)$.

THEOREM 7. *For any given growth bounded graph G and for any fixed integer $k \geq 2$, there is a distributed algorithm that computes a k-domatic partition of size at least $\lceil (\delta_{k-1}+1)/c_k \rceil$ for some constant c_k, in $O(\log \Delta \cdot \log^* n)$ rounds using messages of size $O(\Delta \cdot \log n)$.*

4. DIRECTIONS FOR THE FUTURE

Of the various directions in which our work can be extended, we enumerate three that we consider most significant.

1. The basic premise of this paper is that, from the point of view of saving energy, computing a k-domatic partition for $k > 1$ may be better than computing a 1-domatic partition. While this seems intuitively reasonable and is supported by some preliminary experiments, this premise needs extensive experimental investigation. Specifically, for practical models of energy usage, for a variety of node distributions, and for different network traffic patterns, how much energy is saved by using 2-domatic partitions or 3-domatic partition versus using a 1-domatic partition? Performing this experimental evaluation is an important part of our plan for future work.

2. Our paper does not solve the problem of devising an $O(1)$-factor approximation algorithm for the domatic partition problem on UDGs. As far as we know, this problem is open. This is probably the most significant algorithmic problem that relates to our work. While a sequential $O(1)$-factor approximation algorithm for UDGs and more general classes of graphs would be significant, from the point of view of wireless sensor networks, it would be even more significant if such an algorithm had a distributed version that ran in constant or polylogarithmic number of rounds.

3. The second algorithm in our paper, k-`DP2` runs in $O(\log^* n)$ communication rounds. This algorithms assumes that nodes know distances to neighbors and these distances induce a metric space of low doubling dimension. The third algorithm, k-`DP3` assumes no access to geometric information and relying only on connectivity information, runs in $O(\log \Delta \cdot \log^* n)$ communication rounds. Can k-`DP3` be improved to run in $O(\log^* n)$ rounds? This question is significant because it focuses on the role of geometric information in devising fast distributed protocols for wireless networks. The running time of k-`DP2` and k-`DP3` is dominated by MIS construction. So the question of whether k-`DP3` be improved to run in $O(\log^* n)$ rounds reduces to the question of whether an MIS can be computed for growth-bounded graphs in $O(\log^* n)$ rounds.

5. REFERENCES

[1] M.A. Bonucelli. Dominating sets and domatic number of circular arc graphs. *Discrete Appl. Math.*, 12, pages 203–213, 1985.

[2] M. Cardei and D. Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11(3):333–340, May 2005.

[3] M. Cardei, D. Maccallum, M. X. Cheng, M. Min, X. Jia, D. Li, and D. Du. Wireless sensor networks with energy efficient organization. *Journal of Interconnection Networks*, 3(3-4):213–229, 2002.

[4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks Journal*, 8(5), 2002.

[5] M. Farber. Domination, independent domination, and duality in strongly chordal graphs. *Discrete Appl. Math*, 7, pages 115–130, 1984.

[6] U. Feige, M. M. Halldorsson, G. Kortsarz, and A. Srinivasan. Approximating the domatic number. *SIAM J. Comput.*, 32(1):172–195, 2002.

[7] S. Fujita On the performance of greedy algorithms for finding maximum r–configurations. In *Proceedings of Korea–Japan Joint Workshop on Algorithms and Computation (WAAC), Seoul National University, Seoul*, pages 92–99, 1999.

[8] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP–Completeness,* W.H. Freeman, San Francisco, 1979.

[9] A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, 2003.

[10] H. Kaplan and R. Shamir. The domatic number problem on some perfect graph families. *Inform. Process. Lett.*, 49, pages 51–56, 1994.

[11] S. Kim, D. Culler, J. Demmel, G. Fenves, S. Glaser, T. Oberhein, and S. Pakzad. Structural health monitering of the golden gate bridge. UC Berkeley, NEST Retreat Presentation, Jan 15, 2004.

[12] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 444–453, 2004.

[13] J. Kratochvil. Regular codes in regular graphs are difficult. *Discrete Math.*, 133, pages 191–205, 1994.

[14] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In *Proceedings of the 19th International Symposium on Distributed Computing (DISC)*, 2005.

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

[16] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

[17] A. Mainwaring, J. Polastre, R. Szewczyk, , and D. Culler. Wireless sensor networks for habitat monitoring. Technical Report IRB-TR-02-006, Intel Research Laboratory at Berkeley, 2002.

[18] M. Marathe, H.B. Hunt III, and S.S. Ravi. Efficient approximation algorithms for domatic partition and on-line coloring of circular arc graphs. *Disc. Appl. Math*, 64, pages 135–149, 1996.

[19] T. Moscibroda and R. Wattenhofer. Maximizing the lifetime of dominating sets. In *Proceedings of the 5th. IEEE International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks*, 2005.

[20] S.L. Peng and M.S. Chang. A simple linear time algorithm for the domatic partition problem on strongly chordal graphs. *Inform. Process. Lett.*, 43, pages 297–300, 1992.

[21] A. Proskurowski and J.A. Telle. Complexity of graph covering problems. *Nordic J. Comput.*, 5, pages 173–195, 1998.

[22] A.S. Rao and C.P. Rangan. Linear algorithm for domatic number problem on interval graphs. *Inform. Process. Lett.*, 33, pages 29–33, 1989.

[23] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE International Conference on Communications*, 2001.

[24] A.P. Sprague. NP-completeness of the domatic number problem on circular arc graphs. In *Proc. ACM Southeast Conf.*, CD-ROM, 1999.

[25] Y. Xu, J. S. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Mobile Computing and Networking*, pages 70–84, 2001.