Energy Conservation in Wireless Sensor Networks via Domatic Partitions *

Sriram V. Pemmaraju Imran A. Pirwani

March 28, 2006

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. 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 unit ball graphs 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. Eulidean 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.

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 [12, 7]. 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.

*Department of Computer Science, The University of Iowa, Iowa City, IA 52242. E-mail`[sriram, pirwani]@cs.uiowa.edu`.

A standard approach for reducing energy consumption is to carefully schedule node activity. As has been observed in [2], 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 [2, 15]. In [2], 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 [15], the problem of rotating the responsibility of being a coordinator has been abstracted as the *domatic partition problem*.

We model the network as a 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. 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 [15] 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, this means that if we use a 2-domatic partition instead of just a domatic partition to schedule nodes, then each node would spend a much smaller fraction of the time being active. However, there is a problem with this assertion. 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 $D^{(2)}$. However, there may be no node of $D^{(2)}$ in u's neighborhood so what is u supposed to do? 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 is offset by the fact it costs each node 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 (Section 2.1) 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 as k grows from 1 to 2 and from 2 to 3 the size of a k-domatic partition grows substantially. 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 $(\delta_{k-1} + 1)/\alpha_k$ for some constant α_k. If we had shown that the size of the computed k-domatic partition is at least $(\delta_k + 1)/\alpha_k$, then that would have been a constant-factor approximation algorithm. However, we do not show this and in fact, for the most general class of graphs we consider, there may not be a k-domatic partition of size $(\delta_k + 1)/\alpha_k$ for any constant α_k. 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 [4, 15] returns a partition of size at least $(\delta + 1)/\log n$. This algorithm also solves the k-domatic partition problem and returns a partition of size at least $(\delta_k + 1)/\log n$. In general, this lower bound and the lower bound of $(\delta_{k-1} + 1)/\alpha_k$ 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* [9]. 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. This model takes into account transmission errors that occur when a pair of nodes are almost at the boundary of each other's transmission range. 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.

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. It is easy to see that for any fixed d, the d-dimensional Euclidean space is a metric space with a constant doubling dimension. We call a UBG, a *doubling UBG* if for some constant ρ, the UBG can be embedded in a metric space of doubling dimension ρ. An arbitrary UBG need not be growth-bounded, but any doubling UBG is indeed growth-bounded (Lemma 1 in [9]). It is also easy to see that a UDG is a special kind of a doubling UBG, one that resides in 2-dimensional Euclidean space. Taking advantage of the geometry of a UBG and the fact that pairwise distances are available, we provide algorithms for UBGs that run faster for doubling UBGs than those that run on growth-bounded graphs. Our interest in doubling UBGs is due to the fact that they significantly generalize UDGs and provide a more flexible model for wireless networks, while retaining some geometric characteristics. The notion of doubling dimension has been introduced in [5] and it has been proposed that latencies of peer-to-peer networks and the internet form a metric of constant doubling dimension [5, 8].

We provide an even faster algorithm for UBGs that reside in a Euclidean space such that the coordinates of the points are known.

In summary, we present the following three distributed algorithms.

1. An $O(1)$ round algorithm that computes a k-domatic partition of size at least $(\delta_{k-1}+1)/\alpha_k$, for some constant α_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 a k–domatic partition of size at least $(\delta_{k-1}+1)/\alpha_k$ 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 $(\delta_{k-1}+1)/\alpha_k$ for growth-bounded graphs. Here Δ is the largest degree of a vertex in the graph.

We also provide simulation results that 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 2.1 this will denote a Euclidean distance and in Section 2.2 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 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.

2.1 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
$$\frac{i}{\sqrt{2}} \le x_v < \frac{i+1}{\sqrt{2}} \text{ and } \frac{j}{\sqrt{2}} \le y_v < \frac{j+1}{\sqrt{2}}.$$
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.

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, $1 \le r \le (\delta_{k-1} + 1)/c_{k-1}$, 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. 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 $r \le (\delta_{k-1} + 1)/c_{k-1}$, 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 $\ge r$, there must be a $w' \in V_\ell$ colored r. Since V_ℓ is a clique, $d(w, w') \le 1$. 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. □

Implications 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, we don't know how to compute a constant approximation of an optimal 1-domatic partition, but we know how to compute a 2-domatic partition whose size is at least 1/16th the size of an optimal 1-domatic partition. Experiments show that this is quite conservative and in general, our algorithm returns 2-domatic partitions that are much larger than the largest possible 1-domatic partion. In the following table, we show results obtained by running k-`DP1` on randomly generated UDGs. The last row of the table shows the size of the 2-domatic partition returned by the algorithm. Note that in all cases, this size is much larger than the maximum possible size of a 1-domatic partition (given by $\delta_1 + 1$). In fact, the size of the 2-domatic partition computed by the algorithm is typically close to the size of an optimal 2-domatic as seen by comparing the last two rows of the table.

n	50	60	100	125	150	175	200	225	250	275
δ_1	7	9.2	15.5	19.2	22.7	30.1	30.8	35.2	40.6	41.6
δ_2	15.9	18.1	32.7	42.1	52.8	62.1	71.0	78.5	86.4	97
size	14.8	16.6	29.6	36.3	41.3	49.7	60.5	70.6	72.3	80.1

Let $D^{(k)}$ be a k-dominating set of the network that is a member of the k-domatic partition returned by Algorithm k-`DP1`. As mentioned in the introduction, a node v may not have any node from $D^{(k)}$ in its transmission range and so v has to increase its transmission range to reach some node in $D^{(k)}$. From the proof of Theorem 1, it can be observed that there is always some node in $D^{(k)}$ that lies in one of the grid cells that intersects $Disk(v, 1)$. Thus, in order to reach $D^{(k)}$, v just has to increase its transmission range to completely contain all of these grid cells. 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.

2.2 Using pairwise distances only

Now we assume that nodes are not aware of coordinates, but can sense distances to neighbors. Recall that in Algorithm k-`DP1` we placed a grid of small enough square cells on the plane and constructed 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 u 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 u and a vertex that attaches itself to u 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 will show that this clique-partition has the bounded density property.

Algorithm: k-DP2

1. Compute a proper vertex coloring χ of G.
2. 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 $G_{1/2}$.
3. Each node $v \in I$ broadcasts $\chi(v)$ to its neighbors.
4. Each node $u \in V(G) - I$ receives at least one color and possibly several. Node u then picks a color c from among those it receives such that c is sent by a neighbor at distance at most 1/2 from u. Denote by L_u, the color picked by u. Node u broadcasts the ordered pair $(\chi(u), L_u)$ to its neighbors.
5. Each node $v \in I$ receives ordered pairs of colors from neighbors and constructs the set:
$$S_v = \{\chi(v)\} \cup \{c \mid (c, \chi(v)) \text{ is received from a neighbor at distance at most } 1/2 \text{ from } v\}.$$
6. Each node $v \in I$ broadcasts $(\chi(v), S_v)$ to its neighbors.
7. Each node $u \in V(G) - I$, on receiving a tuple (L_u, S) from a neighbor at distance $\leq 1/2$, sets $S_u \leftarrow S$.
8. Each node $v \in V(G)$ colors itself r, where r is the rank of $\chi(v)$ in the sorted set S_v. Let the new coloring be denoted by χ'. Note that this new coloring is not necessarily proper.

Correctness. In Step (1) of Algorithm k-DP2 a proper $(\Delta(G) + 1)$-vertex coloring of G is computed. This coloring provides locally distinct identifiers to nodes and will become useful when cliques are formed in later steps in the algorithm. In Step (2), 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. 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 (3), the vertices in I, announce their colors to all neighbors. In Step (4) of Algorithm k-DP2 each node u may receive several colors from neighbors. However, it is not the case that u has two neighbors w, w' in $G_{1/2}$ such that u receives the same color from both w and w'. This is because $|ww'| \leq 1$ and therefore w and w' are neighbors in G and cannot have the same color. Thus, when u chooses a color c in Step (4), it is choosing a unique neighbor in $G_{1/2}$ to attach to. We denote u's choice of a color 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 $L_u = \chi(v)$. More precisely, let

$$group(v) = \{v\} \cup \{u \mid L_u = \chi(v) \text{ and } u \text{ is a neighbor of } v \text{ in } G_{1/2}\}.$$

Note that for all $v \in I$, $group(v)$ induces a clique in G and therefore no two vertices in $group(v)$ are assigned the same color by χ. When u broadcasts $(\chi(u), L_u)$ in Step (4), it is announcing its intention to join L_u's group. In Step (5), each node $v \in I$ constructs a set S_v containing the colors of the nodes in $group(v)$. In Step (6), node v tells all members in its group what the set S_v is and in Step (7) all members in $group(v)$ receive this information. The situation after Step (7) can be summarized as follows.

Lemma 1 *For each node $v \in I$, $group(v)$ induces a clique. Each node $v \in I$ constructs a set $S_v = \{\chi(u) \mid u \in group(v)\}$. For each node $u \in group(v)$, $S_u = S_v$.*

After Step (7), members in $group(v)$ already have distinct colors and the only thing left to do is "palette reduction," that is, a recoloring so that members in $group(v)$ are assigned distinct colors from the set $\{1, 2, \ldots, |S_v|\}$. This happens in Step (8).

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. □

Together, the two lemmas above and the proof of Theorem 1 gives us the following result.

Theorem 2 *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.*

Running time. Steps (1)-(2) of Algorithm k-`DP2` takes $O(\log^* n)$ rounds each via the results in [10]. In that paper, a deterministic algorithm running in $O(\log^* n)$ rounds is given for the problem of computing an MIS and for the problem of computing a $(\Delta + 1)$-coloring on a doubling UBG. 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.

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 relied 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. In summary, we have the following result.

Theorem 3 *For any $k \geq 2$, Algorithm k-*`DP2` *computes a k-domatic partition of size at least $(\delta_{k-1} + 1)/c_{k-1}$ in a given UBG in a metric space of constant doubling dimension in $O(\log^* n)$ rounds.*

2.3 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. Using only connectivity information, it is not clear how to quickly compute a clique-partition that has the bounded density property. 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 at most 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 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. 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 $d(u, v) \leq 2\}$. Compute a proper vertex coloring of G^2. Denote this coloring by χ.
2. Compute an MIS I of G.
3. 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'.
4. Each node $u \in V(G) - I$ sets a variable $status_u \leftarrow available$. Each node $v \in I$ sets a variable $status_v \leftarrow unavailable$.
5. 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 color, $\chi(u)$, to neighbors.
    (b) Each node $v \in I$ colored r by χ' receives a color from each available neighbor and constructs the set $C_v = \{\chi(u) \mid u \in N(v)$ and $status_u = available\}$.
    (c) Each node $v \in I$ colored r by χ' picks the smallest $\lfloor \delta_1/(L+1) \rfloor$ colors from C_v and places these in S_v. Node v then broadcasts $\{\chi(v)\} \cup S_v$ to neighbors. 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 colors from a neighbor in I. Node u then checks if $\chi(u) \in S$ and if so u sets $status_u \leftarrow unavailable$ and $S_u \leftarrow S$.
6. Each unavailable node v computes the rank r of $\chi(v)$ in S_v and colors itself r. Each available node colors itself 1. Let the latest coloring of vertices be denoted χ''. Note that this vertex coloring need not be proper.

Correctness. In Step (1) of `2-DP3` a proper vertex coloring χ of G^2 is computed. For any vertex $v \in V(G)$, no two vertices in $N[v]$ are assigned the same color by χ. This property will be used in later steps of the algorithm. In Step (2), an MIS I of G is computed. In Step (3), the subgraph H of G^2 induced by I is formed and a proper $(\Delta(H)+1)$-vertex coloring χ' of H is computed. Thus each vertex $v \in I$ has two colors, $\chi(v)$ and $\chi'(v)$. The following lemma shows that the number of colors used by χ' is bounded above by a constant. It is true that χ restricted to I is also a proper vertex coloring of H, however there is no reason to believe that χ uses a constant number of colors for H.

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 $J = \{Disk(v', 1/2) \mid v' \text{ is a neighbor of } v \text{ in } H\}$ are pairwise disjoint and are all contained in $Disk(v, 5/2)$. Therefore, $|J| < \frac{\pi(5/2)^2}{\pi(1/2)^2} = 25$. □

After Step (3), each node $u \in V(G) - I$ is available for attaching itself to a neighbor $v \in I$. This is indicated in Step (4) 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 (5). In particular, for each $r = 1, 2, \ldots, L + 1$, all nodes in I colored r execute Steps 5(b)-(c) in parallel. In Step 5(a) all available nodes announce their availability to neighbors by broadcasting their colors. In Step 5(b) each node $v \in I$ colored r receives colors from available nodes and places these in a set C_v. Note that no two colors received by v are the same and therefore there is a bijection between C_v and available neighbors of v. In Step 5(c), each node $v \in I$ colored r picks $\lfloor \delta_1/(L + 1) \rfloor$ colors 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 5(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 $v' \in I$, $v' \neq v$. 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 $\{\chi(v)\} \cup S_v$ is broadcast by v to all neighbors. In Step 5(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 $\{\chi(v)\} \cup S_v$. The following lemma summarizes the situation immediately after Step (5).

Lemma 6 *For any $v \in I$, define $group(v) = \{v\} \cup \{u \in N(v) \mid \chi(u) \in S_v\}$. Then for all $v \in I$, $group(v)$ induces a subgraph of G of diameter ≤ 2 and $|group(v)| \geq (\delta_1 + 1)/(L + 1)$.*

Proof: Let v be an arbitrary vertex in I. The fact that $group(v)$ induces a subgraph of G of diameter ≤ 2 is obvious. Since $|S_v| = \lfloor \delta_1/(L + 1) \rfloor$ and since χ assigns distinct colors to $N[v]$,

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

□

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 (5) 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.

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 (6) 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, $1 \leq r \leq \lceil (\delta_1 + 1)/(L + 1) \rceil$, v is dominated (and hence 2-dominated) by D_r. □

Running time. Step (1) requires the construction of a proper vertex coloring of G^2. There are several deterministic distributed algorithms that return a proper vertex coloring of an arbitrary n-vertex graph in $O(\log^* n)$ time [11, 14], using a number of colors that is polynomial in Δ. Any of these can be used for Step (1). Using the result in [9] we can compute an MIS of the given UDG G, even without any location or distance information, in $O(\log \Delta \cdot \log^* n)$ time. The rest of the algorithm consists of 2 local broadcasts and some local computations. Therefore the overall running time of the algorithm if $O(\log \Delta \cdot \log^* n)$.

The algorithm `2-DP3` is well-defined for arbitrary graphs. In the proof of correctness of the algorithm, Lemma 5 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 other two lemmas in the proof of correctness of `2-DP3` follow from Lemma 5 and therefore hold for bounded growth graphs as well. The algorithm `2-DP3` also runs in $O(\log \Delta \cdot \log^* n)$ time for bounded growth graphs. This is because the result in [9] that shows how to compute an MIS in $O(\log \Delta \cdot \log^* n)$ time, holds for bounded growth graphs. This discussion implies the following theorem.

Theorem 4 *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, in $O(\log \Delta \cdot \log^* n)$ rounds.*

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 $|V_i| \geq (\delta_{k-1} + 1)/\alpha_k$ for some constant α_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 $\lceil (\delta_{k-1} + 1)/\alpha_k \rceil$. Constructing a k-uniform partition can be done in a manner very similar to what is described in `2-DP3`. We end this section with the following theorem.

Theorem 5 *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 $\lceil (\delta_{k-1} + 1)/\alpha_k \rceil$ for some constant α_k, in $O(\log \Delta \cdot \log^* n)$ rounds.*

References

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

[2] 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.

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

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

[5] Anupam Gupta, Robert Krauthgamer, and James 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.

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

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

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

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

[14] Gianluca De Marco and Andrzej Pelc. Fast distributed graph coloring with o(&dgr;) colors. In *SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 630–635, 2001.

[15] 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.

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