Maximizing the lifetime of a sensor network through domatic partition

Kamrul Islam
School of Computing
Queen's University
Kingston, Canada
Email: islam@cs.queensu.ca

Selim G. Akl
School of Computing
Queen's University
Kingston, Canada
Email: akl@cs.queensu.ca

Henk Meijer
School of Computing
Queen's University
Kingston, Canada
Email: henk@cs.queensu.ca

Abstract

Distributing sensing and data gathering tasks to a dominating set is an attractive choice in wireless sensors networks since it helps prolong network lifetime by engaging such a subset of nodes for these tasks and letting other nodes go into energy-efficient sleep mode. Because they are busy all the time for sensing, processsing, and transmitting data, nodes in the dominating set quickly run out of energy. One possible way to overcome this situation is to find a number of dominating sets among the nodes of the network and use them one by one iteratively. In this paper, we investigate the problem of finding the maximum number of disjoint dominating sets called the domatic partition problem in unit disk graphs. Although the domatic partition problem is NP-hard in general graphs, it is unknown whether the same is true for unit disk graphs. However, we present an algorithm towards solving this problem (approximately) together with experimental results and give a conjecture based on our results about the maximum number of disjoint dominating sets in unit disk graphs.

1. Introduction

Consider a set of sensors deployed in the plane where each sensor has the same transmission range which is normalized to 1. We model the sensors by a geometric graph $G = (V, E)$ where the node set V represents the sensors and the edge set E consists of links where a link between sensors exists if their Euclidean distance is ≤ 1. A subset $D \subseteq V$ is a dominating set if every vertex $v \in V \setminus D$ has at least a neighbor in D. In this paper, we consider the problem of generating subsets $D_1, D_2, \cdots, D_k$ such that each $D_i \subseteq V$ is a dominating set and $D_i \cap D_j = \phi$, $i \neq j$, and the objective is to maximize k. Dominating sets play an important role in saving energy in individual sensors in wireless sensor networks which will be clear from the following discussion. In general it is expected that the size of a dominating set be as small as possible.

Once deployed with a finite amount of energy, nodes in sensor networks continue to spend energy sensing, gathering, and distributing data from the environment. Since the nodes (sensors) cannot be replenished with energy, one viable option is to prolong the network by carefully utilizing the energy of individual nodes for these tasks. In order to achieve this goal, one obvious way is to exploit the physical proximity of nodes. Since closely positioned sensors in the plane sense almost similar data, it is important to make active as few sensors as possible from such nearby nodes, and put the rest of the nodes in a sleep mode to conserve energy. Thus, by keeping only a few nodes active for gathering, processing, and transmitting data we save valuable energy for other nodes. This naturally leads us to utilize the concept of a dominating set where the nodes selected for carrying on the sensing tasks on behalf of their nearby sensors form the dominating set. The smaller the size of the dominating set the better is the energy saving since more sensors can be put in the energy-efficient sleep mode.

However, this solution has one major drawback in terms of balanced energy distribution among the sensors. This is because the sensors in the dominating set will quickly consume their energy by sensing and transmitting data while the other inactive sensors (that do not belong to the dominating set and are in the sleeping mode) can save their energy. This will result in disproportionate energy consumption among the sensors which is not expected in many applications where network lifetime depends on the functioning of individual sensors. Even finding the minimum dominating set, that is, the one with the fewer number of nodes, will be of no help in this scenario.

This problem can be alleviated if we replace an existing dominating set D_i with another disjoint dominating set D_j in the network and place the nodes in D_i in sleeping mode. This facilitates maximizing the network lifetime since the active role of is now shifted to a completely new set of nodes. Thus the basic idea is to find as many disjoint dominating sets as possible and use each one for a certain amount of time and then replace it with another one and so on. If initially all the nodes start with the same amount of energy, then one can expect that the energy consumption will likely be balanced among the nodes. The maximum number of disjoint dominating sets is bounded by the minimum degree of the graph plus one, since a node can either be in one of the dominating sets or be dominated by at least

one of its neighbors. A graph which possesses the maximum number of disjoint dominating sets given by the bound is called domatically full. An example of a domatically full graph is given in Figure 1(a). However, not all graphs are domatically (see Figure 1(b)) full.

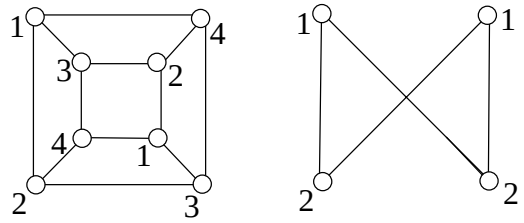

Figure 1. (a) In this 3-connected graph (it is a general graph not a UDG), we have four disjoint dominating sets. Each dominating set is denoted by the same digits. Hence the domatic partition of this graph is 4. (b) The graph does not admit more than two disjoint dominating sets although the minimum degree plus one is 3.

If we can have a family of $D_1, D_2, \cdots, D_k$ disjoint dominating sets in $G = (V, E)$, where $D_i \subseteq V$, then we can employ the first dominating set D_1 for a certain time interval, say, $[0, t_1]$. Thereafter, we can replace D_1 by D_2 and make D_2 active for the $[t_1, 2 * t_1]$ time period, and so on until we are finished with D_k being active for $[(k-1) * t_1, k * t_1]$ time interval in which case we can repeat the whole process starting with D_1. It is clear that the larger the value of k, the longer we can prolong the lifetime of the network. However, as mentioned above, the optimal (maximum) value of k can be at most $\delta + 1$, where δ is the minimum degree in G. In this paper, we investigate the problem of finding the maximum number of disjoint dominating sets (called the domatic partition problem), that is, the maximum value of k (called the domatic partition number)in the special case of unit disk graphs.

2. Related Work

The minimum dominating set problem is one of the classical graph theory problems that has been extensively studied a [3], [17] and proven to be NP-hard in general. For general graphs, one can show [3] that unless $NP \subseteq DTIME\ (n^{O(\log\log n)})$, no polynomial time algorithm can approximate the minimum dominating set problem better than $\ln\triangle - o(\ln\triangle)$, where $\triangle$ is the maximum degree of the graph. In the context of unit disk graphs where two nodes with maximum Euclidean distance 1 have an edge, the dominating set still remains as NP-hard [4]. However, it can be approximated within a constant factor. In [2], Alzoubi et al. show that the size of any maximal independent set in a unit disk graph is at most 4 times the size of the optimal dominating set. Since a maximal independent set is a dominating set, this proves that a dominating set is at most 4 times the size of the optimal dominating set. The dominating set problem has also been studied in distributed settings. In a distributed environment, a constant factor approximation ratio is obtained in [6] in O($\log n^2$) time. Luby [7] gives a randomized distributed algorithm to find a constant approximation to the minimum dominating set problem in O($\log n$) time.

However, all the aforementioned algorithms focus on finding a single 'small' sized dominating set in the graph. the domatic partition problem is one of the problems proved to be NP-hard in [3] for general graphs. This problem has been throughly studied by Feige et al [11] who provide some useful insights about the problem for general graphs. In [11], the authors prove that every graph with maximum degree $\triangle$ and minimum degree δ contains a domatic partition of size $(1-o(1))(\delta+1)/\ln\triangle$. They show that $1/(\ln\triangle)$ is the best possible approximation ratio for this problem for general graphs unless $NP \subseteq DTIME\ (n^{O(\log\log n)})$. A natural greedy approach to this problem is to find small dominating sets in order to maximize the number of disjoint dominating sets in a graph. Fujita [12] studies several variants of greedy algorithms and shows that their performance ratio is no better than $1/\sqrt{n}$.

In [9], the authors look at a variant of the domatic partition problem and propose randomized approximation algorithms for maximizing the network lifetime. The problem they consider is called the Maximum Cluster-Lifetime problem where they aim to find an optimal schedule consisting of a set of pairs $(D_1, t_1), \cdots, (D_k, t_k)$ where D_i is a dominating set and t_i is the time during which D_i is active. That is, D_1 is active in the interval $[0, t_1]$ and D_i is active in the interval $[\sum_{j=0}^{i-1} t_j, \sum_{j=0}^{i} t_j]$. The lifetime of a schedule is defined as $L(S) = \sum_{i=1}^{k} t_i$. The problem then asks for the schedule S with maximum length $L(S)$ such that $\sum_{i:v\in D_i} t_i \leq b_v$ where b_v is the maximum time node v can be in a dominating set. The randomized algorithms computes a schedule which is O($\log n$) approximation to the Maximum Cluster-Lifetime problem with probability at least $1 - o(1/n)$.

The authors in [13] propose a centralized algorithm, without providing any worst case analysis or bound, to generate a number of disjoint dominating sets in order to cover a geometrical region by network nodes for as long as possible. A general version of the domatic partition called k-domatic partition ($k \geq 2$) where in each dominating set a node is dominated by at least k nodes is considered in [10] where the authors propose three deterministic distributed algorithms. Each of the algorithms constructs a $k-$ domatic partition of size at least a constant fraction of the largest possible $(k-1)$-domatic partition.

The rest of the paper is organized as follows. In Section 3 we provide definitions and assumptions that are used throughout the paper. The algorithm is presented in Section 4. We provide a theoretical analysis of our algorithm in Section 5 followed by experimental results in Section 6. We conclude in Section 7.

3. Model and Definitions

We assume that sensors are deployed in the plane and model a sensor network by an undirected graph $G = (V, E)$ where the vertex set V denotes the set of sensors and E represents the links $(u, v) \in E$ between two sensor nodes $u, v \in V$ if they are within their transmission range (recall that every sensor node $u \in V$ has the same transmission range normalized to 1). A subset $D \subseteq V$ is a dominating set if every vertex $v \in V \setminus D$ has at least a neighbor in D. Our algorithm to find disjoint dominating sets operates in rounds where at each round i we construct a dominating set D_i, where $D_i \cap D_j = \phi, i \neq j$ and $i, j \geq 1$. We color the nodes in D_i with color class i.

Define the neighbor sets $N(u)$ and $N[u]$ of sensor node u as $N(u) = \{v|(u, v) \in E, u \neq v\}$ and $N[u] = N(u) \cup \{u\}$ (also known as u's *closed* neighborhood). By $N_k(u)$ we mean the set of nodes from u which are at most k hops away. For simplicity we use $N(u) = N_1(u)$. The degree $deg(u)$ of a node is the number of neighbors it has, that is, $deg(u) = |N(u)|$. By $\delta = min_{v \in V} deg(v)$ we mean the minimum degree in the graph. Each node u is identified by a unique index denoted as $id(u)$.

3.1. Compatible and Incompatible nodes

A node $v \in N[u]$ is called an uncovered neighbor of u at iteration (round) i if there is no node $w \in D_i$ that is a neighbour of v (we call it v is not covered by any node in D_i). The total number of uncovered neighbors of u (including u) in iteration i is denoted as, $uncov_i(u) = |\{v|v \in N[u], vw \notin E, \forall w \in D_i\}|$. The *dom* number of a node u denoted by $dom(u)$ is the number of neighbors (including u) that belong to some dominating sets, that is, $dom(u) = |\{v|v \in N[u], v \in D_j, j \geq 1\}|$. Each node u maintains a triple at round i, $t_i(u) = \prec 1/uncov_i(u), dom(u), id \succ$ which is updated in each round. If $uncov_i(u) = 0$, then we assign $uncov_i(u) = 1/(n + 1)$ where $n = |V|$. We call a node u *compatible* if $uncov_i(u) \neq 1/(n+1)$, else it is called *incompatible*. For any two compatible nodes, u and v we say u is smaller than v in round i if $t_i(u)$ is lexicographically smaller than $t_i(v)$ and denote the order by $t_i(u) < t_i(v)$. Note that only compatible nodes are compared.

4. Algorithm for the Domatic Partition Problem

In this section we present a description of our algorithm. Our algorithm works in rounds starting from round 1 and in each round i we compute a dominating set D_i such that $D_i \cap D_j = \phi, i \neq j, i, j \geq 1$. The algorithm runs for $\delta + 1$ rounds. However, it stops in round j if we cannot form a new disjoint dominating set D_j or we have $j = \delta + 1$. If $j = \delta + 1$ then we have reached the optimal (maximum) number of disjoint dominating sets. Otherwise, $j - 1$ is the the maximum number of disjoint dominating sets $D_1, D_2, \cdots, D_{j-1}$ returned by our algorithm.

Let $D'_1 = \phi$. Starting at round 1, first, we compute a triple for each node in G and select the smallest node among the compatible nodes. From the triple it is easy to see that we always find such a single node. Let u be the node. We color u 1, add u to D'_1 (note that D'_1 is a temporary set that grows as dominating nodes are added to it). We update $t_1(v)$, $v \in N[u]$. Then we pick the smallest compatible node in $V \setminus D'_1$, color it 1, insert into D'_1 and update the nodes' triples. At each selection of a node u in D'_1 we make at least one node incompatible. We repeat the process until all nodes become incompatible. Assign $D_1 = D'_1$. We claim that nodes in D_1 form a dominating set. At this stage $\forall_{u \in G} uncov_i(u) = 1/(n + 1)$ and $\forall_{v \in G \setminus D_1} dom(v) \geq 1$.

We now describe in general how a dominating set D_i is formed at round $i \geq 2$. First, initialize $\forall_{u \in G} uncov_i(u) = deg(u)$ and let $D'_i = \phi$. (As before D'_i is a temporary set used in the i-th round to contain the dominating nodes that are added to it as the algorithm progresses and when D'_i becomes a dominating set we rename it as D_i). Select the smallest node among the compatible nodes in $V \setminus (D_1 \cup D_2 \cdots D_{i-1} \cup D'_i)$, color it i, add to D'_i. Update the triples of the corresponding nodes. Then find the smallest compatible node in $V \setminus (D_1 \cup D_2 \cdots D_{i-1} \cup D'_i)$, color it i and insert into D'_i and update the triples of the nodes whose $uncov_i(.)$, $dom(.)$ values are changed. Continue this process until one of following cases occurs

(I) all nodes become incompatible

or

(II) $V \setminus (D_1 \cup D_2 \cdots D_{i-1} \cup D'_i) = \phi$ and there exists at least one compatible node in G

or

(III) $V \setminus (D_1 \cup D_2 \cdots D_{i-1} \cup D'_i) \neq \phi$ and there exists a node u such that $\forall u' \in N[u]$, $u' \in D_1 \cup D_2 \cdots D_{i-1} \cup D'_i$.

If Case (I) is satisfied then assign $D_i = D'_i$ and we have already constructed the i-th dominating set D_i and next round $i + 1$ is invoked to generate D_{i+1}.

On the other hand, if $V \setminus (D_1 \cup D_2 \cdots D_{i-1} \cup D'_i) = \phi$ and there is at least a compatible node (Case (II)) then we cannot form the current dominating set and the algorithm reports that $i-1$ is the maximum number of disjoint dominating sets and the execution ceases. This is because there is no node to be included in D'_i and some nodes are still uncovered. Similarly if Case (III) is satisfied then the algorithm terminates and reports $i-1$ is the maximum number of disjoint dominating sets. This is because there is no neighborhood $N[u]$ of u from which we can select a node to cover u.

Let j represent the maximum number of dominating sets returned by our algorithm, that is, the algorithm constructs $D_1, D_2, \cdots, D_j$ disjoint dominating sets.

Algorithm for Domatic Partition

Input: A unit disk graph $G = (V, E)$

Output: j disjoint dominating sets $D_1, D_2, \cdots, D_j$

1: Compute $t_1(u) =\prec 1/uncov_1(u), dom(u), id \succ$, $\forall u \in V$
2: $D'_1 = \phi$;$i = 1$
3: **while** $i \leq (\delta + 1)$
4: Find the smallest compatible node u in $V \setminus (D'_1 \cup D'_2 \cup \cdots \cup D'_i)$
6: **if** there is such u **then**
7: $D'_i = D'_i \cup \{u\}$
8: **else**
9: **if** Case (II) or Case (III) is satisfied **then**
11: $j = i - 1$
12: break
13: **else**
14: **if** all nodes are incompatible **then**
15: $D_i = D'_i$
16: $i = i + 1$
17: $j = i$
18: $D'_{i+1} = \phi$
19: **endif**
20: **endif**
21: **endif**
22: update $\forall u \in V t_i(u) =\prec 1/uncov_i(u), dom(u), id \succ$
23: **endwhile**

Figure 2. Algorithm for the domatic partition problem.

4.1. Algorithm Illustration

We illustrate the execution of our algorithm through the example shown in Figure 4. Nodes are marked in lowercase letters below and integers s on the left of nodes denote the dominating sets D_s in which the nodes belong. In the first iteration, we find node a is the smallest compatible node ($t_1(a) =\prec 1/uncov_1(a), dom(u), id(a) \succ =\prec 1/8, 0, id(a) \succ$) in the graph, where $uncov_1(.) = deg(.)$. Select a, add to D'_1 and we update triples $t_1(.)$'s of nodes whose $uncov_1(.)$ and $dom(.)$ values have been changed. After this update, we compute the smallest compatible node in $V \setminus D'_1$ and find b since b's triple $t_1(b) =\prec 1/5, 0, id(b) \succ$ is the smallest. Putting b in D'_1 and repeating the process we compute node c as the smallest compatible node in $V \setminus D'_1$ and insert into D'_1. At this point we update the triples and find that all nodes have become incompatible which puts an end to round 1 and we obtain the first dominating set $D_1 = D'_1 = \{a, b, c\}$.

In the second round, we initialize $\forall_{u \in G} uncov_2(u) = deg(u)$ and the $dom(u)$ values are kept from the previous iteration and $D'_2 = \phi$. Note that all nodes now have $dom(.) = 1$. In this round, we find d since $t_2(d) =\prec 1/6, 1, id(d) \succ$ is the smallest compatible node in $V \setminus (D_1 \cup D'_2)$ and add it to D'_2. After updating the triples we find the smallest compatible nodes e, f, g in order in $V \setminus (D_1 \cup D'_2)$ and insert them in D'_2. Note that e is chosen before f because although they have the same $uncov_2(e) = uncov_2(f) = 3$ and $dom(e) = dom(f) = 1$ values, we assume $id(e) < id(f)$. This round terminates now since all nodes are incompatible. This round produces the second dominating set D_2 by assigning $D_2 = D'_2 = \{d, e, f, g\}$.

We proceed to round three with $D'_3 = \phi$ and following the algorithm we obtain the smallest compatible nodes h, i, j, k in order in $V \setminus (D_1 \cup D_2 \cup D'_3)$ and put them in D'_3. We stop with the third disjoint dominating set $D_3 = D'_3 = \{h, i, j, k\}$ as the nodes in G become incompatible. As we move to round four, ($D'_4 = \phi$) we first find ℓ and m as the smallest compatible nodes and add them to $D'_4 = \{\ell, m, n\}$. But then $V \setminus (D_1 \cup D_2 \cup D'_3) = \phi$ and there exists a number of compatible nodes in G, namely g, k etc. Since the algorithm cannot construct the fourth dominating set D_4, it abandons D_4 and produces as final output the disjoint dominating sets D_1, D_2, and D_3.

5. Theoretical Analysis

In this section we present an analysis of our algorithm. First we show that in each round m we compute a dominating set D_m where $m \leq j$ (recall j is the maximum number of disjoint dominating sets returned by the algorithm) and $D_m \cap D'_m = \phi$, $m, m' \leq j$ and $m \neq m'$.

Lemma 5.1: In G, D is a dominating set if and only if there is no compatible node in G.

Proof: We prove the lemma by contradiction. Assume that D is a dominating set and there exists at least a compatible node v in G. This means either v or some of its neighbors are not covered by any node $w \in D$. Then there can be two situations, either (i) only v is uncovered or

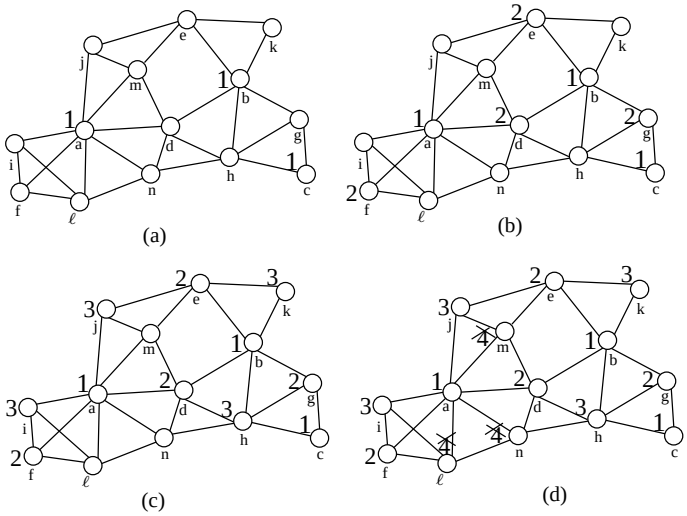

Figure 3. Disjoint dominating sets computation. D_1, D_2, and D_3 are shown in (a), (b), and (c) respectively. (d) D_4 cannot be formed.

(ii) more than one node in $N[v]$ is uncovered. In any of the cases D is not a dominating set in G, a contradiction.

Now we prove the other direction. Again, assume for contradiction that all nodes are incompatible and D is not a dominating set in G. By definition, if all nodes are incompatible then each of them is dominated by some nodes in D contradicting the fact that D is not a dominating set in G. □

In the following, our algorithm guarantees that it can always generate the first dominating set in a connected unit disk graph.

Lemma 5.2: For D_1 only Case (I) is satisfied (Case (II) and Case (III) are never reached by the algorithm while computing D_1). Hence $j \geq 1$ (the number of dominating sets returned by the algorithm) (Line 11 in the algorithm).

Proof: Assume for contradiction that Case (II) is reached while D'_1 (recall that D'_1 is a temporary set which is assigned to D_1 when D'_1 becomes a dominating set in G) is being computed. Since Case (II) is satisfied, $V \setminus D'_1 = \phi$ and we have at least one compatible node. But this is a contradiction since if $V \setminus D'_1 = \phi$, D'_1 is already a dominating set and there cannot be any compatible nodes in G.

Similarly assume Case (III) is satisfied while computing D'_1. Then $V \setminus D'_1 \neq \phi$ and there is node u such that $\forall u' \in N[u]$, $u' \in D'_1$. This is a contradiction because the situation shows that u as well as all its neighbors belong to D'_1 and hence D'_1 is a dominating set.

It is easy to see that Case (I) must be satisfied if Case (II) or Case (III) is not encountered. Thus the smallest value of j can be at least one and hence for Line 11 $j \geq 1$. □

As our main objective is to generate disjoint dominating sets, in the following we show that the algorithm can indeed produce dominating sets in G such that any pair of dominating sets are mutually disjoint. That is, there are no two dominating sets in $D_1, D_2, \cdots, D_j$ which has a common node.

Lemma 5.3: Each D_p is a dominating set and $D_p \cap D_q = \phi$ for $1 \leq p < q \leq j$.

Proof: First we prove that D_p is a dominating set by contradiction. Suppose D_p is not a dominating set in round p. Then there is some node v not dominated by any nodes in D_p. That is, $uncov_p(v) \geq 1$ which implies that v is a compatible node. This is a contradiction (from Lemma 5.1).

Now we show that $D_p \cap D_q = \phi$ for $1 \leq p, q \leq j$ and $p \neq q$. Without loss of generality assume $q > p$ and $D_p \cap D_q \neq \phi$. Therefore there exists at least a node in $D_p \cap D_q$. Let $u \in D_p \cap D_q$. Assume $M = V \setminus (D_1 \cup D_2 \cdots \cup D_p)$ where dominating sets $D_1, D_2, \cdots, D_p$ have already been computed. In computing D_q in the q-th iteration we select nodes from M. Thus if u is picked in D_q then either u does not belong to D_p or D_p is an empty set, yielding a contradiction. □

In the following we show the time complexity of our algorithm.

Lemma 5.4: The time complexity of the algorithm is O(n^3).

Proof: The while loop in the algorithm runs for $\delta + 1$ times which could be as big as O(n). For each iteration of the loop, to find the smallest compatible node it takes O(n) time. Also in each iteration of the loop, for each node that is inserted in the current dominating set we update the triples of all its neighbors and doing so takes O(n^2). Thus the time complexity of the algorithm is O(n^3). □

6. Simulation Results

We conducted extensive simulations on random networks to study the performance of our proposed algorithm. In this section, we provide and analyze experimental results regarding domatic partitions produced by our algorithm and compare with the optimal (maximum) domatic partitions. In our experiments, first we generate unit disk graphs G by placing nodes uniformly and randomly and test the connectivity of G and compute the minimum degree of G. If the underlying graphs are connected we apply our algorithm to construct disjoint dominating sets and analyze its performance in these average graphs.

6.1. Experimental Setup

Experiments are done using our own simulator in Java. In order to evaluate the average performance critically, we have chosen square fields of different sizes namely, 600m x 600m, 500m x 500m, 400m x 400m, and 300m x 300m. For

each of these squares, $n \in \{100, 200, 300, \cdots, 900, 1000\}$ nodes are randomly distributed. The transmission range of all nodes are set to 80m. For each of the squares and for each value of n we generate 100 different topologies, and run our algorithm which generates a number of disjoint dominating sets. The choice of different field sizes for the same input size helps generate relatively sparse (for larger squares) and dense graphs (for smaller squares) to evaluate our algorithm in both situations.

6.2. Experimental Results

Setting the transmission range to 80m for all nodes, we apply our algorithm in a 600m x 600m square with $n = 100$ and compute the domatic partition (DP) number (i.e., the number of disjoint dominating sets) and compare with the optimal DP number. As mentioned before, the optimal DP number is the minimum degree of G plus one. Then we generate 100 random graphs successively and for each graph we compute and compare these DP numbers (one by our algorithm and the other is the optimal value). In order to compute the optimal DP value we use the upperbound of the optimal value which is the minimum degree of G plus 1. Finally, we obtain the average DP number by our algorithm and compare it with the average optimal DP number of all the 100 random topologies we generated. Then we increment the value of n by 100 (i.e., $n = 200$) and repeat the whole procedure. We continue incrementing the value of n by 100 each time and repeating the experiment until $n = 1000$.

To obtain more convincing results and to see how the results vary in terms of the DP number, we apply our algorithm and perform the same procedure in squares of sizes 500m x 500m, 400m x 400m, and 300m x 300m, varying n from 100 to 1000 as before and generating 100 different topologies for each value of n. Changing the sizes of squares affects the degrees of nodes, that is, the smaller the area of the field the denser the graphs are and hence the larger the minimum degrees of the graphs. Intuitively our experiments show that for smaller field sizes the DP numbers are greater than that of larger fields for the same values of n. The results are plotted in Figure 4 where the x-axis shows the number of nodes in the generated topology and the y-axis represents the average DP number returned by our algorithm (shown as non-solid lines, marked as 'Alg' at the top-left corner of the figure) and the average optimal (maximum) DP number (shown as solid lines, marked as 'Opt' at the top-left corner of the figure). As can be understood from the figure, the bottom line pair (solid and non-solid line) shows the average results of our algorithm and the optimal DP numbers when the input field size is 600m x 600m. For smaller values of $n = 100 - 400$, the average DP number almost coincides with the optimal and the values are around $3 - 5$. For higher values of $n = 900 - 1000$, the DP number by our algorithm and optimal numbers are in between $9 - 12$. Other pairs of lines are similarly shown with their respective field sizes. In these average cases our experimental results are close to the optimal for all values of $n = 100 - 1000$ and in all different sizes of fields. It may be worth mentioning that for smaller value of $n <= 400$ the maximum difference between our average DP value and the optimal is 2 (additive factor) and for larger values of n the maximum difference is 3 (additive factor).

6.3. Cardinalities of Dominating Sets

We also provide some interesting results regarding the average number of nodes in each of the dominating sets we generate. For each run of the algorithm to compute the DP number for a certain value of n, we keep track of the minimum and maximum sizes of dominating sets among the disjoint dominating sets. Since we generate 100 topologies for each value of n we compute the averages of the minimum and maximum sizes of dominating sets.

In Figure 5 and 6 we plot these values, that is, in the $x-$axis we have the number of nodes and in the y-axis we show the average sizes (the average minimum size in Figure 5 and the average maximum size in 6) of dominating sets. In general the results we obtain show that, for unit disk graphs, the denser the graphs are the smaller the cardinalities of dominating sets. In an intuitive sense, this happens since in a denser graph each node has relatively many neighbors and when it becomes a dominating node it can dominate many nodes. For example, in Figure 5, for a 300m x 300m field, the average minimum size of dominating sets varies from $8 - 10$ for all values of n used in our experiments whereas for a 600m x 600m square the average minimum size fluctuates from 16 to 35. Similar results are shown in Figure 6 for the average maximum size of dominating sets. A sample output of our algorithm is shown in Figure 7.

6.4. Conjecture about the Maximum Number of Disjoint Dominating Sets

Unit disk graphs have a nice property that for any node in the graph, there can be at most five independent nodes in its neighborhood. In other words, for a node u we need at most 5 nodes to cover (dominate) all its neighbors. This result can be further strengthened. In a constant neighborhood (e.g., a k-hop neighborhood) unit disk graphs allow to have at most a constant number of independent nodes. This is a strong property of unit disk graphs and other *growth-bounded* graphs (which also have the same property). Utilizing this property and depending on the experimental results we have the following conjecture:

Conjecture 6.1: For unit disk graphs, the domatic partition problem can be solved within a constant factor of the optimal.

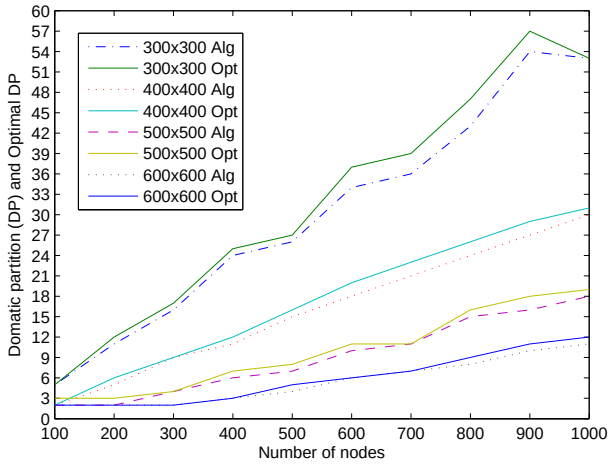

Figure 4. Showing the comparison between the average domatic partition (DP) number returned by our algorithm and the average optimal DP number (minimum degree of G plus 1) in different field sizes for each value of $n \in \{100, 200, \cdots, 1000\}$.

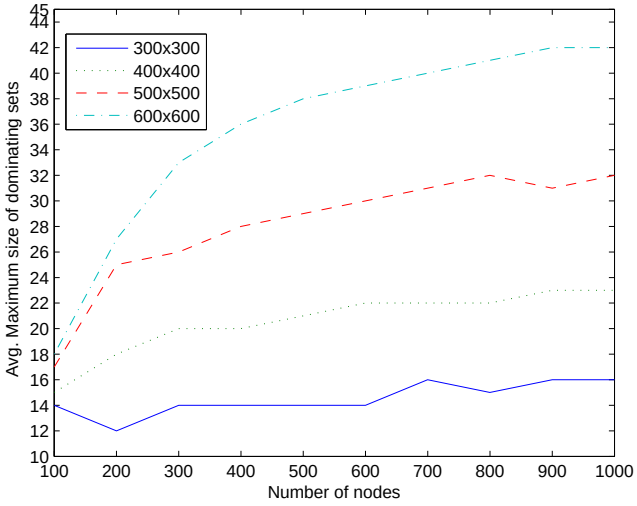

Figure 6. Showing the average maximum cardinality of dominating sets found by our algorithm for different values of n.

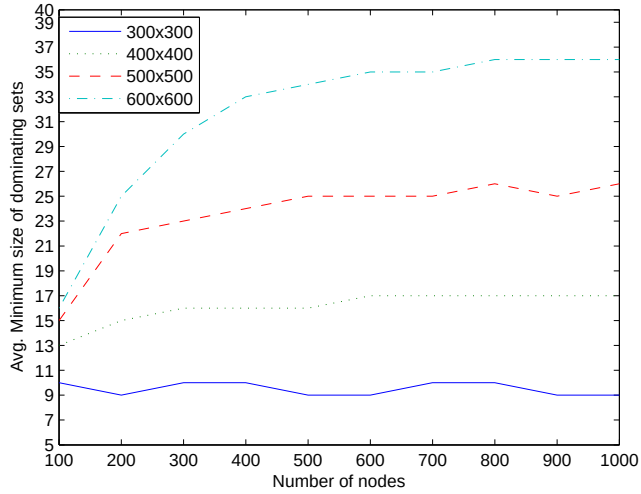

Figure 5. Showing the average minimum cardinality of dominating sets found by our algorithm for different values of n.

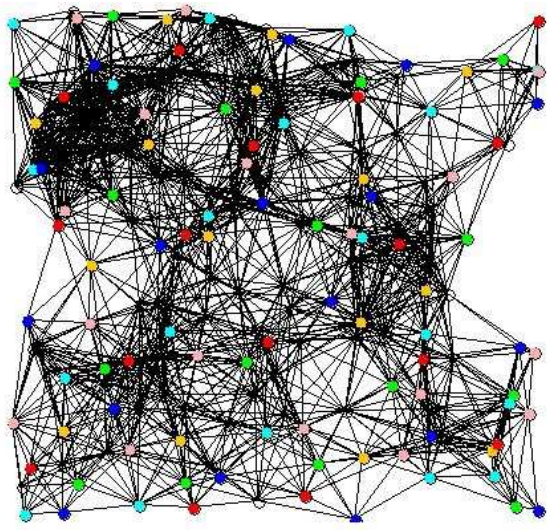

Figure 7. A sample output with 200 nodes distributed on 400m x 400m square field where both the optimal DP and the computed DP numbers are 6 (shown in 6 different colors).

7. Conclusions

In this paper, we have presented an algorithm based on domatic partition for maximizing the lifetime of wireless sensor networks. We have provided extensive simulation results to assess the performance of our algorithm where the experiments provide good results. However, one intriguing open question is to come up with an algorithm that gives an approximation factor (preferably 'small') for unit disk

graphs. We believe that a constant factor algorithm is achievable by utilizing the structural properties of unit disk graphs. For example, a node can have at most five independent nodes in its neighborhood. For unit disk graphs, the only known factor for this domatic partition problem is logarithmic and obtained by a randomized algorithm [9]. Since centralized algorithms do not scale up for large networks, distributed algorithms are sought for large wireless sensor networks. Thus it would be challenging to find a distributed solution with a small constant approximation factor for the domatic partition problem in unit disk graphs.

References

[1] J. Schneider and R. Wattenhofer, "A Log-Star Distributed Maximal Independent Set Algorithm for Growth-Bounded Graphs," In *Proc of PODC*, 2008.

[2] K. M. Alzoubi, P.J. Wan, and O. Frieder, "Message-optimal connected-dominating-set construction for routing in mobile ad hoc networks," In *Proc MobiHoc* 2002.

[3] M. Garey and D. Johnson, *Computers and intractability, a guide to the theory of NP-completeness*, W. H. Freeman and Company, 1979.

[4] B. Clark, C. Colborn, and D. Johnson, "Unit disk graphs," *Discrete Mathematics*, 86:165-177, 1990.

[5] F. Kuhn and R. Wattenhofer, "Constant time distributed dominating set approximation," In *Proc. of PODC*, 2003.

[6] F. Kuhn, T. Moscibroda and R. Wattenhofer, "What cannot be computed locally," In *Proc. of Mobicom*, 2004.

[7] M. Luby, "A simple parallel algorithm for the maximal independent set problem," *SIAM Journal on Computing*, 15:1036-1053, 1986.

[8] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, No. 1, January 2002.

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

[10] S. Pemmaraju and I. Pirwani, "Energy conservation in wireless sensor networks via domatic partitions," In *Proc. of MobiHoc* 2006.

[11] U. Feige, M. Halldorsson, G. Kortsarz, and A. Srinivasan, "Approximating the domatic number," *SIAM Journal of Computing*, 32(1): 172-195, 2003.

[12] S. Fujita, "On the performance of greedy algorithms for finding maximum r-configurations," In *Proc. of WAAC*, 1999.

[13] S. Slijepcevic and M. Potkonjak,"Power efficient organization of wireless sensor networks," In *Proc. of Infocom*, 2001.

[14] R. Gupta, J. Walrand, and O. Goldschmidt, "Maximal cliques in unit disk graphs: polynomial approximation," In *Proc. INOC 2005*, Portugal, 2005.

[15] S. Kutten and D. Peleg, "Fast distributed construction of small k-dominating sets and applications," *Journal of Algorithms*, 28:40-66, 1998.

[16] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*, SIAM, Philadelphia, PA, 2000.

[17] R. Karp, "Reducibility among combinatorial problems," In *Proc. of a symposium on the complexity of computer computations*, pp 85-103, 1972.