Distributed Generation of a Family of Connected Dominating Sets in Wireless Sensor Networks

Kamrul Islam, Selim G. Akl, and Henk Meijer

School of Computing, Queen's University,
Kingston, Ontario, Canada K7L 3N6
{islam,akl,henk}@cs.queensu.ca

Abstract. We study the problem of computing a family of connected dominating sets in wireless sensor networks (WSN) in a distributed manner. A WSN is modelled as a unit disk graph $G = (V, E)$ where V and E denote the sensors deployed in the plane and the links among them, respectively. A link between two sensors exists if their Euclidean distance is at most 1. We present a distributed algorithm that computes a family S of $S_1, S_2, \cdots, S_m$ non-trivial connected dominating sets (CDS) with the goal to maximize $\alpha = m/k$ where $k = \max_{u \in V} |\{i : u \in S_i\}|$. In other words, we wish to find as many CDSs as possible while minimizing the number of frequencies of each node in these sets. Since CDSs play an important role for maximizing network lifetime when they are used as *backbones* for broadcasting messages, maximizing α reduces the possibility of repeatedly using the same subset of nodes as backbones. We provide an upper bound on the value of α via a *nice* relationship between all minimum vertex-cuts and CDSs in G and present a distributed (localized) algorithm for the α maximization problem. For a subclass of unit disk graphs, we show that our algorithm achieves a constant approximation factor of the optimal solution.

Keywords: Wireless Sensor Network, Maximal Clique, Distributed Algorithm, Minimum Vertex-cut.

1 Introduction

The connected dominating set is one of the earliest structures proposed as a candidate for a virtual backbone in wireless networks [6, 11]. Given a graph $G = (V, E)$, where V and E are sets of nodes and edges respectively, a subset D of V is a *dominating set* if node $u \in V$ is in D or adjacent to some node $v \in D$. A subset of V is a CDS if it is a dominating set and induces a connected subgraph. A minimum cardinality CDS is called an optimal CDS (or simply OPT). Henceforth we use the words 'optimal' and 'minimum' interchangeably. In the context of sensor networks, a well studied problem is that of finding an OPT in a unit disk graph (UDG) which is often used to model connectivity in sensor networks where an edge between two nodes exists if and only if their Euclidean distance is at most 1. In this paper, we use UDGs to model sensor networks and take G to be a UDG.

1.1 A Motivating Example

We are motivated by an important observation for the general *CDS* construction in wireless sensor networks where the the main goal is to obtain a "small" approximation factor for the computed *CDS*, since finding a minimum cardinality *CDS* is NP-hard [5]. Consider the wheel graph in Figure 1 (a) with 9 nodes. Assume that each node has 9 units of energy and each transmission for a unit of data from a node to its neighbor consumes one unit of energy, neglecting the consumption of energy for message reception. The optimal *CDS* contains only node 9 at the center. Suppose each border node $\{1, 2, \cdots, 8\}$ has a unit of data to be broadcast via the center node 9 in each communication round. That is, whenever a node on the periphery sends a message, it is node 9 which broadcasts (relays) it and all other nodes receive the message. After eight rounds corresponding to eight border nodes transmitting their data through the center node, the center node will be left with 1 unit of energy while all the border nodes still contain eight units of energy each. After the 9th round, node 9 will have completely exhausted its energy and the other nodes will have 8 units of energy each for continuing operation. Thus finding an optimal solution might not always be helpful in scenarios like this and it is not hard to construct a family of such examples.

One easy way to get around this problem is to devise algorithms that can in some way evenly distribute the energy consumption among the nodes in the network by not using the same subset of nodes for all broadcasting operations. Clearly this may not yield a 'small-sized' or constant factor approximation solution to the *CDS* problem but can be used to prolong the lifetime of the network which depends on the functioning of every node in the network. Using node 9 as a *CDS* for one broadcasting operation then leaving it free and engaging six consecutive nodes from the border nodes as next *CDS*s for successive rounds (See Figure 1(b), (c)) for each broadcasting clearly enhances the overall network lifetime. In other words, we have two scenarios. First, we can use only the middle node as the *CDS* which gives $m/k = 1$, where m is the number of *CDS*s generated and k is the maximum number of occurrences of a node in these *CDS*s. In the second scenario, we can use 3 *CDS*s where the nodes are taken from the border nodes only and one *CDS* containing only the middle node. Thus we get $m/k = 4/3$ which means that if the network is used for 4 rounds, for the first scenario, the middle node is used for 4 rounds, and in the second scenario, no node is used for more than 3 rounds.

In general, the above problem can be solved by finding a *large* number of *disjoint connected dominating sets* since doing so has a direct impact on conserving the energy of individual sensors and hence prolonging the network lifetime. However, as we will show, it is not always possible to find a large number of such disjoint sets because the number of disjoint *CDS*s is bounded by the size of a minimum vertex-cut of the graph. Thus the main focus of this paper is to find as many *CDS*s as possible while minimizing the frequency of participation of each node in the network. In other words, we would like to maximize $\alpha = m/k$ which would increase

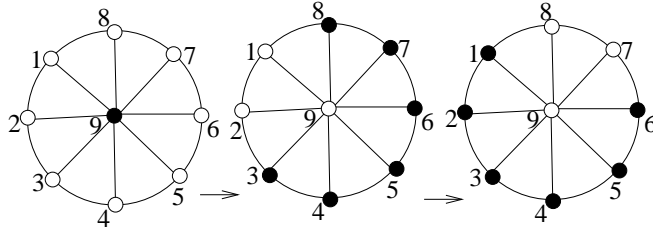the overall network lifetime by incurring less energy consumption on the part of each node for broadcasting.

Fig. 1. Family of *CDS*s

We provide a novel distributed algorithm (that requires the knowledge of nodes at most 3-hops away) to compute a family of *CDS*s with a goal to maximize α. Our approach for obtaining such a family of sets depends on the computation of maximal cliques using only the one-hop neighborhood of each node combined with a simple coloring scheme for the nodes that requires each node to know information of its neighbors at most 3-hops. Maximal cliques in *UDG*s can be found in polynomial time in *UDG* [5, 19].

2 Related Work

There is a large body of literature [1–3, 7–9, 12, 14] regarding finding the *MCDS* in wireless sensor networks which focuses on the construction of a *single CDS* and the effort is to devise algorithms to produce a small approximation factor for it. Aside from looking for a single *CDS* with a small approximation factor, there has been some research towards finding a family of dominating sets [15–18] for data gathering and sensing operations in sensor networks. In [17], the authors consider the problem of finding a maximum number of disjoint dominating sets called the *domatic partition*. They show 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 provide a centralized algorithm to produce a domatic partition of size $\Omega(\delta/\ln\triangle)$. The papers [16, 15, 18] are somewhat related to ours since they investigate the problem of maximizing the lifetime of wireless networks by generating disjoint dominating sets (not connected dominating sets) for data gathering purposes. The general focus is that the nodes in the dominating set act as "coordinators" for the network and the rest of the nodes remain in an energy-efficient *sleep* mode. Thus to maximize the lifetime of the network, it is important to rotate the coordinatorship among the nodes in the network so that every node is roughly given an equal opportunity to go into sleep mode.

In particular, the authors in [18] 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 [16]. Their algorithms construct a $k-$ domatic partition of size at least a constant fraction of the largest possible $(k-1)$-domatic partition.

In [15], the authors look at a variant of the domatic partition problem and propose a randomized algorithm for maximizing the network lifetime. The problem considered in [15] 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 (DS) and t_i is the time during which D_i is active. 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 that v can be in a DS. The randomized algorithm computes a schedule which is an O($\log n$)-approximation with high probability.

Our approach differs from those mentioned above in the way that we look to find as many *CDS*s as possible with the aim to minimize the number of times each node participates in the network. To the best of our knowledge our distributed algorithm for generating such multiple *CDS*s for broadcasting is the first of its kind. The rest of the paper is organized as follows. In Section 3 we provide definitions and assumptions that are used throughout the paper. The distributed algorithm is presented in Section 4. A theoretical analysis of our algorithm is presented in Section 5 followed by conclusions in Section 6.

3   Preliminaries

We assume that sensors are deployed in the plane and model a sensor network by an undirected graph $G = (V, E)$ where V is the set of sensors and E represents the set of links $(u, v) \in E$ between two sensor nodes $u, v \in V$ if they are within their transmission range. We assume that every sensor node $u \in V$ has the same transmission range which is normalized to 1. We 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 $N[u] = N(u) \cup \{u\}$.

Each node u has an ordered triple, $(ct(u), deg(u), id(u))$ where the first element $ct(u)$ denotes the number of times u has participated in *CDS*s, $deg(u)$ and $id(u)$ denote the degree and id of u. Initially, $ct(u) = 0$, $\forall u \in V$, and this is updated as the algorithm is executed. For a maximal clique $X \subseteq N[u]$ and $u, v \in X$, we say node u is lexicographically smaller than node v if $(ct(u), deg(u), id(u)) < (ct(v), deg(v), id(v))$, i.e., either $ct(u) < ct(v)$ or $ct(u) = ct(v)$ and $deg(u) < deg(v)$ or $ct(u) = ct(v)$ and $deg(u) = deg(v)$ and $id(u) < id(v)$. The rank of node u with respect to clique X denoted by $r(X(u))$ is the index of the lexicographically sorted (ascending order) triples of the nodes of X. Note that the rank $r(X(u))$ can be changed by the update of the variable $ct(u)$. Our algorithm is local which can be defined as follows: An algorithm is called $k - localized$ if

each node is allowed to exchange messages with its neighbors at most k−hops away and take decisions accordingly based on this information. We assume a synchronized message passing system as in [10, 15, 16] in which time is divided into equal slots. In each communication slot a node of the network graph is capable of receiving messages from its neighbors, performing local computations, and broadcasting a message to its neighbors. The time complexity of an algorithm is the number of time slots it needs to produce the desired result. In principle, the message size can be arbitrarily large. Although we use a synchronous system, it is well known that a synchronous message passing algorithm can be turned into an asynchronous algorithm with the same time complexity, but with a larger message complexity. Sensors know their geographic locations and obtain their neighbors' locations, degrees, ids and $ct(.)$ values through exchanging *"Hello"* messages.

3.1 Problem Formulation

Given a connected UDG $G = (V, E)$, we would like to find a family S of subsets $S_1, S_2, \cdots, S_m$ such that

i) $\forall i$, S_i is a CDS
ii) and $\alpha = m/k$ is maximized where $k = \max_{u \in V} |\{i : u \in S_i\}|$.

4 A 3-localized CDS Algorithm

We provide a distributed (localized in a strict sense) algorithm for generating a family of CDSs. The algorithm uses a simple coloring scheme where initially all sensors are white and when a sensor becomes a member of the CDS S_i, it becomes black. Set S_i, whose members are all black becomes the virtual backbone and remains active for a *round* (consisting of fixed units of time) to perform broadcasting. After the round, all black nodes change their color into white. Then part of the algorithm is executed again and a new set S_{i+1} is produced which remains active for a round and so on. In this section, we present a simple algorithm for computing a family of CDSs in G which is based on coloring the nodes of G. Here we outline the algorithm. The algorithm consists of two phases described in what follows.

4.1 Dominating Set Generation: Phase I

In Phase I, each node u computes a maximal clique in $N[u]$ following the polynomial time algorithm [19] which computes maximal cliques in UDGs. Let $C(u)$ denote the maximal clique computed by u. Henceforth maximal cliques are just called cliques unless otherwise stated. Node u sends $C(u)$ to all its direct neighbors v if $v \in C(u)$. Similarly u receives a clique $C(v)$ from $v \in N(u)$ if $u \in C(v)$. Denote the set of all such cliques received by u as $C(N(u))$, i.e., $C(N(u)) = \bigcup_{u \in C(v), v \in N(u)} C(v)$. For each clique $C^+ \in (C(u) \cup C(N(u)))$, u computes its rank $r(C^+(u))$

which gives u, $|C(u) \cup C(N(u))|$ ranks in total. For each C^+, the smallest ranked node of C^+ becomes active by coloring itself black. This yields a set of black nodes B in G and we prove in the next section that B forms a dominating set. Now we need to add a subset of nodes C' from the remaining white nodes in G and color them black to make B a *CDS*, i.e., $B \cup C'$ becomes a *CDS*. In Phase II, we determine such set C'.

This phase of the algorithm which is executed at each node u, is given in Figure 2.
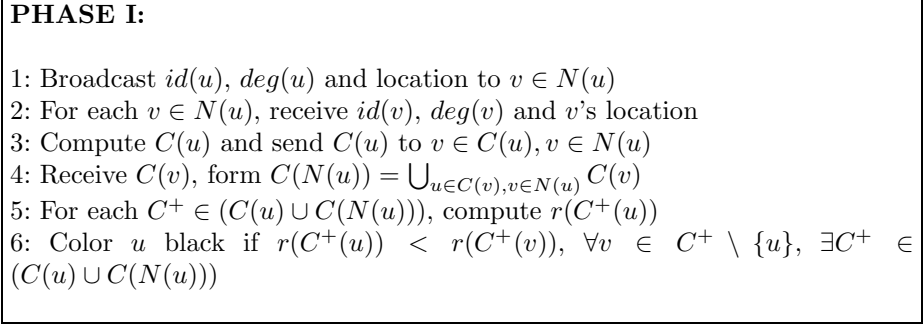

PHASE I:

1: Broadcast $id(u)$, $deg(u)$ and location to $v \in N(u)$
2: For each $v \in N(u)$, receive $id(v)$, $deg(v)$ and v's location
3: Compute $C(u)$ and send $C(u)$ to $v \in C(u), v \in N(u)$
4: Receive $C(v)$, form $C(N(u)) = \bigcup_{u \in C(v), v \in N(u)} C(v)$
5: For each $C^+ \in (C(u) \cup C(N(u)))$, compute $r(C^+(u))$
6: Color u black if $r(C^+(u)) < r(C^+(v))$, $\forall v \in C^+ \setminus \{u\}$, $\exists C^+ \in (C(u) \cup C(N(u)))$

Fig. 2. Phase I

4.2 Connected Dominating Set Generation: Phase II

Recall that Phase I finds a dominating set B (consisting of black nodes) in G. In Phase II we compute a connected dominating set, that is, we select a set of connectors C' to connect the nodes in B such that $B \cup C'$ becomes a *CDS*. Since the distance between any two closest nodes in B can be at most three, we divide this phase into two cases where in Case 1 we discuss how two nodes in B which are two hops away from each other connect themselves via connectors. Case 2 is the same but such nodes are exactly three hops away from each other.

Case 1: Let $B(u, i)$ (resp. $W(u, i)$) be the set of all black (resp. white) nodes which are exactly i hops away from u, including u. The i hop distance between u and v is the minimum number of hops between them. If u is black then u sends $B(u, 1)$ to all the white nodes $W(u, 1)$. Node $c \in W(u, 1)$ is called a *connector* if it connects two black nodes by turning itself black. Node $c \in W(u, 1)$ receives $B(u, 1)$ from each black member $u \in B(c, 1) \setminus \{c\}$ and computes whether $\bigcup_{u \in B(c,1) \setminus \{c\}} B(u, 1)$ forms a connected induced subgraph. Let $B^*(c) = \bigcup_{u \in B(c,1) \setminus \{c\}} B(u, 1)$. Node c sends a 'POS' (positive) message to $B(c, 1)$ if $B^*(c)$ is connected. Otherwise, c sends $B^*(c)$ to $W(c, 1)$ and receives $B^*(c')$ from $c' \in W(c, 1)$. If there is no c' such that $B^*(c) \subset B^*(c')$ (strict subset) then c issues a 'NEG' (negative) message to $B(c, 1) \setminus \{c\}$, otherwise c refrains from sending any messages. However, if $B^*(c) = B^*(c')$ for all c' then c sends a 'NEG' message.

However, if there is at least a 'NEG' message u receives from its white neighbors, $W(u,1)$, then u forms a list $L(u)$ consisting of all received messages along with the corresponding ids of the senders. For example, if u receives 'POS' messages from nodes with id 12 and 4 and a 'NEG' from nodes 2 and 9 then $L(u) = \{(v_{12}('POS'), v_4('POS'), v_2('NEG'), v_9('NEG')\}$. Node u sends $L(u)$ to its neighbors exactly 2 hops away. In the same way, u receives $L(z)$ from $z \in B(u,2)$. If $L(u) \cap L(z)$ has at least one node with a 'POS' message then they are connected via some black nodes. Otherwise, u and z select the lexicographically smallest connector c by sending c a 'YES' message. After receiving 'YES' from u and z, c turns into black and joins the current CDS. See Figure 3 (a) and (b) for an illustration.
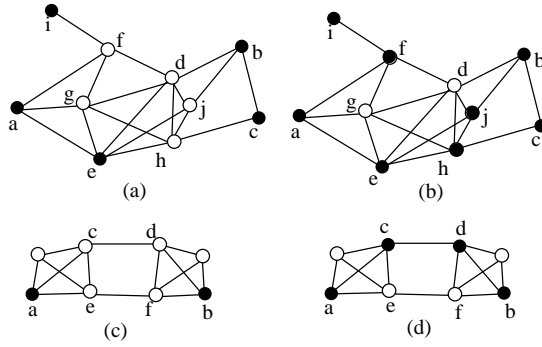

Fig. 3. (a) Case 1: $B^*(h) = B^*(d) = B^*(j) = \{a, e, b, c\}$, $B^*(g) = \{a, e\}$, and $B^*(f) = \{a, e, i\}$. Only $h, d, j,$ and f will each generate a 'NEG' message to their black neighbors while g sends a 'POS' message to a and e. For a and e, $L(a) \cap L(e)$ has a common sender g with a 'POS' message. They do nothing to find connectors. Nodes b and e choose a node from $\{d, j\}$, while e and c take h as connectors which turn into black by receiving 'YES' from from b, c and e. (b) After execution of Case 1. (c) Case 2: a and b are 3-hops away. (d) They select c and d as their average $ct(.)$ value is smaller than that of e and f.

Case 2: Let $P_{uu'}$ denote a shortest path between $u, u' \in B'$ where $u' \in B(u,3) \setminus \{u\}$ and u' is the lexicographically smaller than u. If there exists a path $P_{uu'}$ consisting of exactly two intermediate black nodes then they do nothing since u and u' are already connected via two black nodes. Otherwise $P_{uu'}$ consists of i) a connector c and a white node w and/or ii) two white nodes w', w'' as intermediate nodes. Nodes u and u' choose w and c if there is only one type (i) path or if there are more than one such paths, then they select the one with nodes having the smallest $(ct(c) + ct(w))/2$. Similarly if there is no type (i) path, they select the path with nodes having the smallest $(ct(w') + ct(w''))/2$. In both situa-

tions ties are broken arbitrarily. This is done when u and u' both send 'YES' message to w in type (i) or to w' and w'' in type (ii) paths. After receiving a 'YES' message, node w for a type (i) path or nodes w' and w'' for a type (ii) path join the current CDS by coloring themselves black. If C' is a set of all such w or w' and w'' then $B \cup C'$ forms a CDS we are looking for. See Figure 3 (c) and (d) for an example.

Thus the execution of Phases I and II forms a CDS S_i which consists of the nodes (black) and these nodes remain active for a certain amount of time (a round) for broadcasting while the white nodes stay in the energy-efficient sleep mode. Phase II is given in Figure 4 and a subroutine (called from Phase II) to change color is provided in Figure 5. After serving for a round, all nodes in the CDS, i.e., black nodes, update their $ct(.)$ values by one, send the updated values to their neighbors and change color into white. All nodes in G then start executing from Line 5 Phase I and then Phase II with their updated triples. This generates a new CDS S_{i+1} consisting of black nodes, which remains active for another round. Then $ct(.)$ values are updated and a new set S_{i+2} is constructed, and so on.
Figure 6 illustrates the construction of a CDS by coloring certain nodes black. The numbering of nodes denotes the id of the corresponding node and the number inside the bracket indicates the nodes' $ct(.)$ values. Initially all $ct(.)$ values are set to zero and they are incremented by one each time the respective nodes participate in the CDS. In this example we can construct six different sets.

5   Theoretical Analysis

In this section we give an overview of the theoretical analysis of our algorithm. First we show that Phase I computes a dominating set and then we prove that Phase II forms a CDS. We analyze some other characteristics of our algorithm and provide a relationship between the CDSs and minimum vertex-cuts of G.

Lemma 1. *The set of black nodes B produced in Phase I forms a dominating set in G.*

Proof. We prove the claim by contradiction. Consider a node u. If it is black or connected to a black node then we are done. Otherwise, all nodes in $N[u]$ are white. Let the clique computed by u be $C(u) \subseteq N[u]$. Since each node in $C(u)$ sorts the ordered triples lexicographically, each of them has a unique rank from $\{1, \cdots, |C(u)|\}$. The node with the least rank in $C(u)$ must be black, a contradiction. $\square$

Lemma 2. *The set of black nodes $B \cup C'$ produced in Phase II forms a CDS in G.*

Proof. Assume the subgraph G' induced by $B \cup C'$ is not connected. Consider two neighboring components of G' each containing a node from B and the shortest distance between them is at most three (since B is a dominating set in G). Let u and v be two such nodes. Each of them

PHASE II:
1: **If** u is black **Then** Send $B(u,1)$ to 1-hop connectors C **Endif**
2: **If** u is white **Then**
3:   **If** $B^*(u)$ is connected **Then** Send 'POS' to $B(u,1) \setminus \{u\}$
4:    **Else** //u sends $B^*(u)$ and receives $B^*(u')$
5:     **If** $\nexists u'$ s.t. $B^*(u) \subset B^*(u')$, $u' \in W(u,1) \setminus \{u\}$ **Then**
       Send 'NEG' to $B(u,1)$ **Endif Endif**
6: **Endif**

7: **If** u is black **Then** Receive msgs from C
8:   **If** msg='NEG' **Then**
9:     Send $L(u)$ to $u' \in B(u,2)$ and receive $L(u')$
10:    Send 'YES' to the lexicographically smallest node v **Endif**
11:   **Endif** call color()

12: **If** $P_{uu'}$ has 2 black nodes **Then** do nothing
13:   **Else** //$u' \in B(u,3)$ and lexicographically smaller than u
14:    **If** $P_{uu'}$ is type(i) path **Then**
15:     Select $w, c \in P_{uu'}$ with the smallest $(ct(c) + ct(w))/2$
16:     **Else** //$P_{uu'}$ is type(ii) path
17:     Select $w', w'' \in P_{uu'}$ with the smallest $(ct(w') + ct(w''))/2$
18:    **Endif**
19:    Send w or w', w'' 'YES' to color them black
20: **Endif** call color()
21: **If** u is black **Then** Remain active for a round
22:   Send $ct(u) = ct(u) + 1$ to $N(u)$ **Endif**
23: **If** u is black **Then** Turn u white **Endif**

Fig. 4. Phase II

subroutine color ()
1: **If** u is white **Then**
2:   **If** it receives 'YES'
3:     Turn into black **Endif Endif**
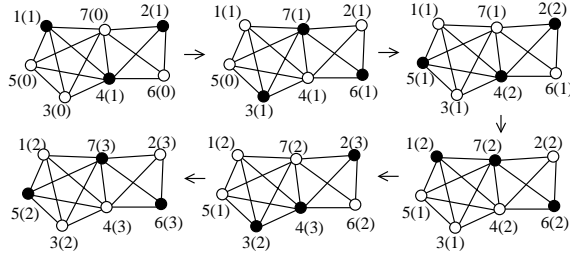
Fig. 5. Subroutine color.

Fig. 6. Illustration of the algorithm to produce a family of *CDS*s.

finds connectors to connect to the nodes in B (nodes at most three hops away from it) which are lexicographically smaller than it. Since lexicographically each node is unique, one of the nodes of u and v must be smaller than the other. Then they must select at most two connectors to join them. If they do not find such connectors then either the shortest distance between them is not at most three or they are already connected by some connectors. In both cases it is a contradiction. □

5.1 Time and Message Complexities

Each node requires only the information of its three-hop neighborhood. Collecting information from a three-hop neighborhood can be done in constant time. Thus the time complexity of the entire algorithm is constant since local computation time (the time spent to compute the cliques in each node's neighborhood) is negligible in synchronous systems [10].

For the message complexity analysis we see that in Phase I each node sends a single message (containing its id, position) to its neighbors, thus generating $O(n)$ messages in total for the network. After computing a clique in its 1-hop neighborhood node u sends this information (id of the nodes in the clique) in a single message to its neighbors and similarly receives cliques from its neighbors. This again makes a total of $O(n)$ messages.

Moreover, $O(n)$ messages are transmitted in Phase II in Lines 1-6 to know about 1-hop connectors. For Lines 7-11 in Phase II, each node sends a single message, and also relays at most 1 message (that is, it receives all messages from its neighbors and merge them into a single message and sends) totaling $O(n)$ messages. For the rest of the algorithm in Phase II, that is, in Lines 12-23, there are $O(n)$ messages generated (a node relays at most two messages and generates its own message). These many messages are required to explore black nodes which are 3-hops away from a certain node. This complexity arises for one single round. Since there can be at most $\triangle$ rounds, the message complexity becomes $O(n\triangle)$. Thus the message complexity of the entire algorithm is $O(n\triangle)$.

5.2   Relation between Minimum Vertex-cuts and *CDS*s

We prove that the maximum value of m is upper bounded by a number that results from a relation between minimum vertex-cuts and any *CDS* (including OPT) in a graph. A *vertex-cut* of G is a set of vertices whose removal renders G disconnected. A vertex-cut of minimum cardinality is called a *minimum vertex-cut*, $\kappa(G)$. Let $\mathcal{K}(G)$ denote the family of all minimum vertex-cuts of G. We show that for any $\kappa(G) \in \mathcal{K}(G)$, $\exists a \in \kappa(G)$ such that $a \in S_i$ where S_i denotes any *CDS* of G. In other words, any *CDS* must take at least one element from each of the minimum vertex-cuts. Instead of "vertex-cut" we use "node-cut" in order to be consistent with the terminology of the paper.

Theorem 1. *In a connected graph G, any CDS (including OPT) must contain at least one node from each $\kappa(G)$ where $\kappa(G) \in \mathcal{K}(G)$ unless G is a complete graph which has no cuts at all.*

Proof. Consider any minimum node-cut $\kappa(G) \in \mathcal{K}(G)$ and suppose for the sake of contradiction that an arbitrary *CDS* S^* of G (S^* can be OPT) does not contain any node from $\kappa(G)$. Let G_1 and G_2 be two connected induced subgraphs generated by the removal of $\kappa(G)$. S^* is a *CDS* and does not contain any node from $\kappa(G)$ so components G_1 and G_2 must each have its own *CDS*, name them $S^*(G_1) \subseteq S^*$ and $S^*(G_2) \subseteq S^*$, respectively. However, $S^*(G_1) \cup S^*(G_2)$ is not connected since the only nodes that could connect $S^*(G_1)$ and $S^*(G_2)$ belong to $\kappa(G)$, thus contradicting the fact that S^* is a *CDS*. □

Here we show a relationship between the number of minimum node-cuts and the cardinality of OPT. The following corollaries are directly deducible from the above theorem.

Corollary 1. *In G, $|OPT| \geq |\mathcal{K}(G)|$ if and only if $\kappa_i(G) \cap \kappa_j(G) = \phi$ for all i, j, $1 \leq i, j \leq |\mathcal{K}(G)|$.* □

Corollary 2. *If G is a tree, then every internal node is a minimum node-cut. Therefore, $|OPT| = |\mathcal{K}(G)| =$ number of internal nodes in G.*
□

Now we have the following lemma.

Lemma 3. *The optimal value of α, α_{opt} is bounded by the size of a minimum node-cut in $\mathcal{K}(G)$.*

Proof. Let M be the set of *CDS*s the optimal algorithm finds in G and $\kappa(G)$ be a minimum node-cut in $\mathcal{K}(G)$. Recall that all $\kappa(G)$s in $\mathcal{K}(G)$ are of the same size, otherwise they would not be minimum node-cuts. Since every *CDS* in M must use at least one node from $\kappa(G)$ (Theorem 1), the maximum value of k (i.e., the maximum number of occurrences of a node in M) will be at least $|M|/|\kappa(G)|$. Hence, $\alpha_{opt} \leq |M|/(|M|/|\kappa(G)|) = |\kappa(G)|$ and the claim follows. □

Now we analyze the performance of our algorithm towards optimizing the value of $\alpha = m/k$ for a subclass of unit disk graphs. We consider a particular subclass of unit disk graphs where each node $u \in V$ belongs to exactly one clique (recall clique means maximal clique). That is, each node computes a clique $C(u)$ but does not receive any clique $C(u') \neq C(u)$ from its neighbors (in other words, $|C(u) \cup C(N(u))| = 1$). There are two constants c and d. For any node u, assume its neighbors belong to at most in c different cliques and for d we define the following. Let the set of neighbors of u which are in cliques $C(u') \neq C(u)$ be $N'(u)$. That is, $N'(u) = \{u''|u'' \in N(u), u'' \in C(u'), C(u') \neq C(u)\}$. Let the number of neighbors of u which are in cliques $C(u') \neq C(u)$ be bounded by d, that is, $|N'(u)| \leq d$. Denote by α_{alg} the value of α obtained by our algorithm.

Lemma 4. *For this subclass (mentioned above) of unit disk graphs $\alpha_{alg} \geq 1/(c*(d+1)) * \alpha_{opt}$.*

Proof. Let there be s cliques computed by the nodes in V and without loss of generality assume all cliques have the same size, $p > 2$. Note that the maximum number of disjoint CDSs possible in this setting can be at most p, implying that the optimal value α_{opt} can be also at most p. This is because generating the $(p+1)$-th CDS must reuse some of the nodes already used in previous CDSs making $k > 1$ (recall k is the maximum number of occurrences of a node in the generated connected dominating sets CDSs). Thus $\alpha_{opt} \leq p$.
In order to find a CDS, our algorithm first selects a node from each of the s cliques in Phase I. Let u be the node selected from clique $C(u)$. Since u's neighbors can be at most in c different cliques, u requires at most $c-1$ *additional* nodes from $C(u)$ that act as connectors to connect to $c-1$ other cliques. Thus the algorithm uses at most c nodes from $C(u)$ for each CDS. Therefore, our algorithm can compute at least $1/c$ CDSs. Now we figure out the value of k, that is, the upperbound of how many times a node participates in different CDS. Since $|N'(u)| \leq d$, u can be selected at most $d+1$ times, once it is selected in Phase I, other times (d times) it is selected (in Phase II) as a connector between two nodes in different cliques. Hence $k \leq (d+1)$. Therefore, $\alpha_{alg} \geq 1/(c*(d+1))*\alpha_{opt}$. □

An example of this subclass is given in Figure 7 with $c = 2$ and $d = 2$.

Theorem 2. *For a complete graph K_n, our algorithm achieves $\alpha_{alg} = \alpha_{opt} = n$.*

Proof. For graph $G = K_n$, initially in Phase I, the algorithm selects the node with the lowest id by coloring it black since all nodes have the same $ct(.)$ value and degree. This node updates its $ct(.)$ value by one. In Phase II, the black node receives all 'POS' messages from other $n-1$ nodes and ignores all such messages and so no connector node is required. Thus we have exactly one node as CDS, i.e., $|S_1| = 1$. In the next round, the second lowest id node is selected and used as the only node in the CDS ($|S_2| = 1$) and so on. Thus we have a family $S_1, S_2, \cdots, S_m$ of CDSs
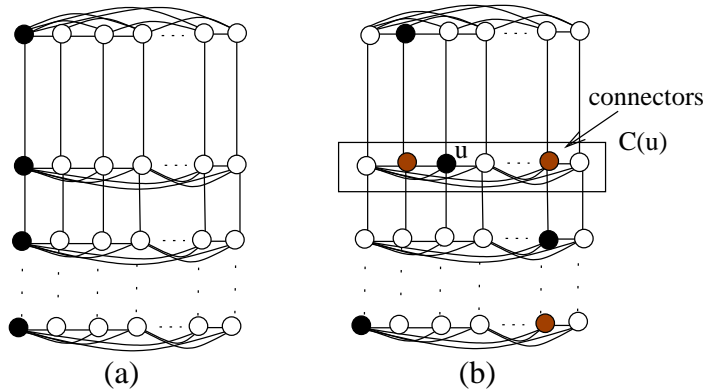
Fig. 7. Illustration of Lemma 4. (a) Optimal algorithm chooses any column of nodes as a *CDS* each time (shown in black) whereas our algorithm (b) first chooses black nodes from each clique and then uses at most two *additional* (shown in brown) nodes from each clique to make them connectors between cliques.

where each set S_i consists exactly of one node of K_n, all nodes in K_n are used in *CDS*s and $S_i \neq S_j$, $i, j \in [1, m]$. This means $m = n$ and $k = 1$. Therefore, $\alpha_{alg} = \alpha_{opt} = n$. □

6 Conclusions

In this paper we have presented a distributed (localized) algorithm for generating a family of connected dominating sets in sensor networks with a view to increasing the lifetime of nodes in the network. Using the same *CDS* (even an optimal one) in sensor networks is not a viable option since the nodes in the set quickly deplete their energy due to their participation in every broadcast. Inspired by this observation, we develop a distributed algorithm which takes care of this situation by generating other connected dominating sets so that the burden of broadcasting can be distributed over the nodes. We show an interesting and important relationship between minimum vertex-cuts and *CDS*s where the cardinality of a minimum vertex-cut limits the number of *disjoint CDS*s. We believe this is the first distributed algorithm of its kind in the context of wireless sensor networks since previous algorithms mainly focus on generating a single *CDS* of small size. Our algorithm obtains a constant factor approximation for a small subclass of unit graphs, however it remains as an interesting open problem to devise an algorithm (centralized or distributed) with a constant approximation factor for unit disk graphs.

References

1. Alzoubi, K. M., Wan, P.J., Frieder, O.: New Distributed Algorithm for Connected Dominating Set in Wireless Adhoc Networks. In: Proc. IEEE HICSS35, 2002.
2. Alzoubi, K. M., Wan, P.J., Frieder, O.: Distributed Heuristics for Connected Dominating Set in Wireless Adhoc Networks: J. on Communication Networks, Vol. 4, No. 1, pp 22-29. March, 2002.
3. Cheng, X., Huang, X., Li, D., Du, D.: Polynomial-time Approximation Scheme for Minimum Connected Dominating Set in Adhoc Wireless Networks. In: Proc. Networks, 2006.
4. Chen, D., Mao, X., Fei, X., Xing, K., Liu, F., Song, M.: A Convex-Hull Based Algorithm to Connect the Maximal Independent Set in Unit Disk Graphs. In: Proc. Wireless Algorithms, Systems, and Applications, pp 363-370. October, 2006.
5. Clark, B., Colborn, C., Johnson, D.: Unit Disk Graphs. Discrete Mathematics. 86:165-177, 1990.
6. Das, B., Bharghavan, V.: Routing in Adhoc Networks Using Minimum Connected Dominating Sets. In: Proc. ICC, pp 376-380, 1997.
7. Funke, S., Kesselman, A., Meyer, U., Segal, M.: A Simple Improved Distributed Algorithm for Minimum Connected Dominating Set in Unit Disk Graphs. ACM Transactions on Sensor Networks Vol. 2, No. 3, pp 444-453. August 2006.
8. Guha S., Khuller, S.: Approximation Algorithms for Connected Dominating Sets. Algorithmica, pp 374-387. 1998.
9. Marathe, M., Breu, H., Hunt, H., Ravi, S., Rosenkrantz,D.: Simple Heuristics for Unit Disk Graphs. Networks, 25, pp 59-68. 1995.
10. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM, Philadelphia, PA, 2000.
11. Sivakumar, R., Das B., Bharghavan, V.: Spine Routing in Adhoc Networks Using Minimum Connected Dominating Sets. In: Proc. ICC, pp 376-380. 1997.
12. Stojmenovic, I., Seddigh, M., Zunic, J.: Dominating Sets and Neighbor Elimination Based Broadcasting Algorithms in Wireless Networks. IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 1, January 2002.
13. Wan, P., Alzoubi, K., Frieder, O.: Distributed Construction of Connected Dominating Set in Wireless Adhoc Networks. In: Proc. Infocom, 2002.
14. Wu, J., Li, H.: On Calculating Connected Dominating Sets for Efficient Routing in Adhoc Wireless Networks. In: Proc. Discrete Algorithms and Methods for Mobile Computing, pp 7-14. 1999.
15. Moscibroda, T., Wattenhofer, R.: Maximizing the Lifetime of Dominating Sets. In: Proc. WMAN, 2005.
16. Pemmaraju, S., Pirwani, I.: Energy Conservation in Wireless Sensor Networks via Domatic Partitions. In: Proc. MobiHoc 2006.
17. Feige, U., Halldorsson, M., Kortsarz, G., Srinivasan, A.: Approximating the Domatic Number. SIAM Journal of Computing, 32(1): 172-195, 2003.
18. Slijepcevic, S., Potkonjak, M.: Power Efficient Organization of Wireless Sensor Networks. In Proc. Infocom, 2001.
19. Gupta, R., Walrand, J., Goldschmidt, O.: Maximal Cliques in Unit Disk Graphs: Polynomial Approximation: In: Proc INOC Lisbon, Portugal, 2005.