

Customizable Localized Computation of Connected Dominating Sets for Self-Organizing Wireless Networks

Jiang Li

Department of Systems and Computer Science
Howard University
Washington, DC 20059, USA

Yingchang Xiang
Rizhao Vocational & Technical College
Rizhao, Shandong 276826
P.R. China

Liran Ma, Fang Liu

Department of Computer Science
The George Washington University
Washington, DC 20052, USA

Qing Xia

Kellar Institute
George Mason University
Fairfax, VA 22030, USA

Abstract

In self-organizing wireless networks, connected dominating sets (CDS) have many applications such as in routing. Many algorithms for localized computation of CDS have been proposed. However, in reality, customization is needed under many situations when certain nodes should be given higher priority for entering CDS. A gap between the research and reality exists because the customizability has not been fully explored so far. In this paper, we attempt to fill the gap by proposing a purely localized algorithm that is highly customizable. Its message complexity is also low. Theoretical analysis is provided for performance verification. We also customize the algorithm to observe better performance. Finally, by simulations, the general (i.e. non-customized) algorithm and the customized version are compared with two purely localized algorithms in term of average CDS sizes.

1 Introduction

As the wireless technology advances rapidly, self-organizing wireless networks such as wireless mesh networks and sensor networks, are on the horizon of ubiquitous application. The infrastructure of such networks is radically different from that of the traditional Internet in that it is formed by the self-coordination among the network nodes after the networks are deployed, whereas the latter is defined along with the deployment. In such an improvised infrastructure, connected dominating sets (CDS) can be leveraged to improve the performance of many functions. For example, because of the improvisation, it is futile to as-

sign dedicated routers prior to deployment. The commonly practiced approach is allowing nodes to decide their own roles (either as a router or a mere sender/receiver) based on the topology known only after deployment. Using those nodes in a CDS (e.g. in [1] [2]) can effectively reduce the number of routers while still being able to provide end-to-end connections all over the networks. Other application areas include media access coordination (e.g. [3]), broadcast (e.g. [4]), energy conservation (e.g. [5]) and topology control (e.g. [6]) etc.

In graph theory, given a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges, a CDS is such a set of vertices that any vertex in V is either in the CDS or is the neighbor of at least one vertex in the CDS. In addition, the subgraph induced by the CDS is connected. In wireless networks, a node can be abstracted as a vertex. An edge is placed between two vertices if the two corresponding nodes can communicate directly. By this mean, a wireless network is abstracted as a graph (usually undirected).

When the size (i.e the number of vertices) of the CDS is minimal, the performance is the best. For example, to save energy in sensor networks and prolong the network life time, only sensors in the CDS are used to forward traffic for other sensors. Obviously, the smaller the CDS, the less energy is consumed, and the longer the network can last. However, finding minimum connected dominating sets (MCDS) is NP-Hard [7]. Researchers have been working on this problem and proposed a number of heuristic algorithms of various performance.

On the other hand, we notice that there has not been an algorithm that provides customizability. In reality, people may want to customize the self-organizing procedure by giving certain network nodes higher priority to join the

CDS. For instance, some nodes could be more powerful, and they are preferred to forward traffic if they are in the right position. In this paper, we address this problem by building such capability into the algorithm. The proposed algorithm has three major features: **1)** It is *purely localized*. Each node only needs information from its neighbors at most two hops away. The time complexity is $O(d + t)$ where d is the average node degree, and t is a smaller value (to be explained in Section 3). **2)** It is *highly customizable*. Users can assign different priorities to different nodes prior to deployment by tuning some parameters, or nodes can decide their own priorities for joining the CDS depending on their situation. **3)** It has low message complexity. The total number of messages sent by each node is at most 3.

Using simulations, we compare the average CDS sizes resulting from our algorithms with two other also purely localized but non-customizable algorithms proposed respectively by Adjih et al [8] and by Wu [9]. We do observe that to achieve the customizability, certain performance is traded off at acceptable loss. Furthermore, we customize the algorithm by considering network topologies more carefully. The purpose is to demonstrate the customizability and to also show that the performance of our algorithm has the potential to be improved by customization. The correctness of the algorithm is proved by theoretical analysis.

At this stage of our research, we only consider those networks with static topology, and we are not yet concerned with the dynamic update of CDS. These problems are to be explored in the future.

In the following, we will first briefly discuss the existing algorithms of CDS computation, followed by the detailed description and theoretical analysis of our algorithm. Simulation results and the conclusions are presented at the end.

2 Related Work

There are many different approaches to construct a CDS. First of all, the algorithms can be categorized as either centralized or distributed. In centralized algorithms (e.g. [10, 11]), a central entity has the global knowledge and controls the construction process. In distributed algorithms, differentiation can be made between those requiring global information (e.g. [2, 12]) and those requiring only local information (e.g. [8, 9]). Although the centralized algorithms and the distributed ones requiring global information usually achieves better performance, they are not scalable. Therefore, we only compare the performance of our algorithm with the localized distributed ones, in which category our algorithm falls in.

The two algorithms MPR [8] and EMPR [9] that we choose for comparison are both *purely localized* in that their time complexity is $O(\Delta^3)$ where Δ is the maximum number of edges emanating from a vertex. Other algorithms

(e.g. [13]), although they also only require local information, their construction processes are actually sequential, and therefore have the time complexity of $O(n)$ where n is the number of vertices in a graph. We do not categorize such algorithms as purely localized.

The MPR algorithm [8] by Adjih et al uses multipoint relays to help CDS construction. Given V as a set of vertices, let $N(V)$ be the union of V and its immediate neighbors. Another vertex set U is covered by V if $U \subset N(V)$. The multipoint relay set $R(v)$ of a vertex v is such a subset of $N(\{v\})$ that covers $N(N(\{v\})) - N(\{v\})$. Each element of $R(v)$ is a multipoint relay of v . To calculate $R(v)$, the following greedy algorithm is used (for convenience, we denote $N_1(v) = N(\{v\}) - \{v\}$ and $N_2(v) = N(N(\{v\})) - N(\{v\})$):

Consider a vertex $u \in N_1(v)$:

Step 1: add those u 's into $R(v)$, if there is a vertex in $N_2(v)$ covered only by u .

Step 2: add u into $R(v)$, if u covers the largest number of uncovered vertices in $N_2(v)$, i.e. the vertices in $N_2(v) - (N_2(v) \cap N(R(v)))$. Repeat this step until all vertices in $N_2(v)$ are covered.

After the MPR sets of all vertices are obtained, two rules are followed for CDS construction. Assuming each vertex has a unique ID, a vertex x is added to the CDS if **1)** x has the smallest ID among $N(\{x\})$ (rule 1), or **2)** x is a multipoint relay of its neighbor y , where y has the smallest ID among $N(\{x\})$ (rule 2).

The EMPR algorithm [9] by Wu is actually an enhanced version of the MPR one. Two enhancements were proposed for calculating MPR sets and the CDS respectively. The enhanced greedy algorithm for MPR calculation is as follows:

Consider a vertex $u \in N_1(v)$:

Step 1: add all free neighbors of v to $R(v)$, where u is a free neighbor of v if the ID of v is not the smallest among $N(\{u\})$.

Step 2: add those u 's into $R(v)$, if there is a vertex in $N_2(v)$ covered only by u .

Step 3: add u into $R(v)$, if u covers the largest number of uncovered vertices in $N_2(v)$, i.e. the vertices in $N_2(v) - (N_2(v) \cap N(R(v)))$. Vertex ID is used to break the tie of coverage if any. Repeat this step until all vertices in $N_2(v)$ are covered.

The other enhancement is for rule 1 in adding vertices to the CDS: a vertex x is added to the CDS if x has the smallest ID among $N(\{x\})$ and has two neighbors that are not adjacent to each other.

With the enhancements, the EMPR algorithm gains some performance improvement over the MPR algorithm. Nonetheless, they both rely on vertex IDs. Although it is possible to replace the IDs with other metrics, the values of

such metrics have to be unique for each node. Moreover, the values and metrics cannot change over time, otherwise the foundation is broken. Therefore, these two algorithms are not fully customizable. Our algorithm described in the following avoids these problems.

3 A Customizable Localized Algorithm of Connected Dominating Set Construction

Our algorithm assumes that each network node has a unique ID. In fact, as long as such ID's are unique within the two-hop neighborhood, the algorithm works fine. We also assume that all node clocks are synchronized when the CDS computation starts.¹

In the following, we will first elaborate the general version of the algorithm, i.e. the one without any customization. After that, a customized version will be provided. For readers' convenience, the most important notifications to be used in the algorithm description and analysis are given in the following table:

Table 1. Notations and their interpretations.

ID_u	Node identifier of node u
$N_i(u)$	The list of i -hop neighbors of node u .
$List_1^u$	The list of dominator neighbors that are <i>at most</i> one-hop away from node u .
$List_2^u$	The list of dominators that are <i>exactly</i> two-hop away from node u .
$List_{invi}$	The list of nodes invited to join the CDS.
S_u	The set of (x, y) such that $x \in List_1^u$, $y \in List_2^u$, and $ID_x < ID_y$.

3.1 The General Algorithm

The algorithm consists of two phases. The first phase (Phase I) builds a maximal independent set (MIS) in which no two nodes are adjacent to each other. The second phase (Phase II) connects all pairs of nodes within the MIS that are two or three hops away. All the nodes within the MIS are termed *dominators* and other nodes *dominatees*. Obviously, each dominatee has at least one dominator as its neighbor. Some dominatees will be elected as *connectors* in Phase II. All dominators and connectors together form a connected dominating set.

In the procedure, nodes broadcast messages for coordination. Three types of messages are involved: *dominator*, *dominatee*, and *connector*, with their

¹This assumption should not be a problem as usually the CDS is computation right after deployment. Node clocks can be easily synchronized before that.

content illustrated in Fig. 1. The *CV* field in all messages accommodates custom values, which can be linked to those operation parameters in the algorithm and make them customizable. The values can be in any form, such as numbers or even function names. Users can assign various values to this fields depending on their needs. The $List_{invi}$ field in the connector message contains all nodes invited to join the CDS such that the sender of the message and its two-hop dominator neighbors in $List_2$ can be connected.

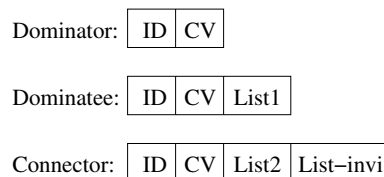


Figure 1. Message formats.

To distributively control the priorities of different message broadcastings, we introduce three time intervals δ_1 , δ_2 , and δ_3 , with $\delta_1 < \delta_2 < \delta_3$. Moreover, we exploit random delays for collision avoidance. A random delay in the range $(0, t)$ for node u is denoted by $Rand_u(t)$. Different instances of $Rand_u(t)$ have different values.

3.1.1 Phase I: Computing an MIS

Starting from the beginning T_0 , Phase I contains $t + 1$ rounds. The first t rounds (rounds $0, 1, \dots, t - 1$), each lasting δ_1 time as shown in Fig. 2, compute a maximal independent set. The last round, with variable length, is used for *dominatee* message broadcastings. A node with an undecided role becomes a dominator with probability $\min(2^i p, 1)$ at round i for $i = 0, 1, \dots, t - 1$, where p is a user-defined parameter that is related to the custom value and network density, and t is chosen such that $2^{t-2} p < 1$ and $2^{t-1} p \geq 1$. Based on this regulation, the probability of being a dominator increases exponentially and all undecided nodes intend to become dominators at round $t - 1$. Note that within each round nodes make decisions probabilistically and independently.

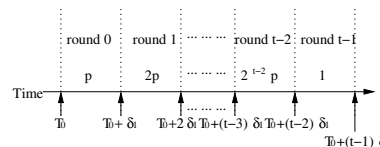


Figure 2. The first t rounds of Phase I.

Initially, the roles of all nodes are undecided. When more and more rounds are executed as time goes by, more and more nodes become dominators or dominatees. The behavior of each node u during Phase I can be illustrated by

the state transition diagram in Fig. 3. The states and transitions are explained in details as follows.

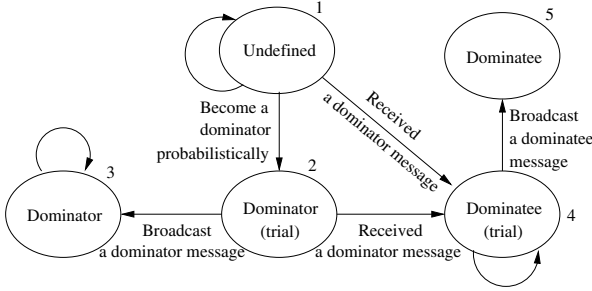


Figure 3. Node state transitions in Phase I.

- **State 1.** Initially all nodes are in state 1. At the beginning of each round i for $i = 0, 1, \dots, t - 1$, a node u determines whether or not to become a dominator with probability $\min(2^i p, 1)$. If not, u stays in state 1; otherwise, it goes to state 2:
 - $1 \rightarrow 1$: u chooses not to become a dominator at the current round;
 - $1 \rightarrow 2$: u chooses to become a dominator at the current round.
- **State 2.** Immediately after the decision of being a dominator is made, u starts a timer set to expire in $Rand_u(\frac{\delta_1}{2})$. This timer is denoted by $Timer_u^1$. When $Timer_u^1$ times out, u broadcasts a dominator message and goes to state 3. If receiving a dominator message before $Timer_u^1$ expires, u goes to state 4. In fact, the timeout value if we use $Rand_u(\tau)$, and link τ to the custom value, while still having it less than $\frac{\delta_1}{2}$.
 - $2 \rightarrow 3$: u 's timer $Timer_u^1$ expires and a dominator message is broadcasted.
 - $2 \rightarrow 4$: u 's timer $Timer_u^1$ is cancelled by a received dominator message. u becomes a dominatee since one of its neighbors has announced its dominator status.
- **State 3.** u becomes a dominator successfully. In the following rounds, u remains in this state.
 - $3 \rightarrow 3$: u does not change its state in Phase I after it goes to state 3.
- **State 4.** u has at least one neighbor that has claimed to be a dominator. Thus u stays in the current state. At the beginning of round t , u starts its timer $Timer_u^1$ set to expire in $Rand_u(\delta_2)$. When $Timer_u^1$ expires, u broadcasts a dominatee message and goes to state 5. Again, similar to the discussion in State 2, the timeout value here can be customizable.

- $4 \rightarrow 4$: u becomes a dominatee since it has at least one dominator neighbor. u stays in this state before round t starts.
- $4 \rightarrow 5$: u successfully broadcasts a dominatee message at round t .

- **State 5.** u has successfully announced its dominatee status to the neighborhood.

Remarks:

- Phase I requires no *a priori* neighborhood information. Each node broadcasts either a dominator or a dominatee message exactly once. At the end of Phase I, a node is either in state 3, or in state 5.
- Limited neighborhood information is collected in Phase I. Nodes are assumed to work in the promiscuous mode, which means that they can extract information from all messages broadcast by their neighbors. Therefore, at the end of Phase I, each node u owns the following information:
 - One-hop neighbors, their status (dominator or dominatee), and their custom values;
 - Two-hop dominator neighbors and the corresponding one-hop neighbors from which to reach the former.
- Even though random delay ($Rand(t)$) is introduced for collision avoidance, it is possible that not all collisions are removed. For simplicity, we assume all nodes can successfully receive messages from their neighbors.
- $Rand(t)$ is customizable if we link the t to the custom value. The algorithm will work fine even if we use values that vary over time for t , as it only changes the order of timeouts. This remark also applies to Phase II in the coming subsection.
- The value of p depends on the custom values of the nodes (and is thus customizable) and the network density. For normal nodes, p can be set to $\frac{1}{d}$ when all nodes are randomly and uniformly deployed, where d is the average degree of the node network. In this case, we have $d = \frac{N}{A} \cdot (\pi r^2)$, where N is the total number of nodes, A is the area of the deployment region, and r is the transmission range of each node. For nodes of higher priority, p can be set higher than $\frac{1}{d}$ so that they are more likely to become dominators.

3.1.2 Phase II: Connecting the MIS

Since round t in Phase I has variable length, it may not be efficient to ask all nodes to launch Phase II at the same time,

even though they are synchronized. For simplicity, we assume each node enters Phase II after a *dominatee* message is received if it is in state 3, or after it enters state 5. We further rely on δ_3 , a multiple of δ_2 , to ensure that a node in Phase II does not broadcast any messages before all neighboring broadcastings for Phase I finishes.

Let D be the maximal independent set computed in Phase I, which contains all dominators at state 3 in Fig. 3. Let $\{D_1, D_2\}$ be any partition of D such that $D = D_1 \cup D_2$ and $D_1 \cap D_2 = \phi$. If none of D_1 and D_2 is empty, the shortest distance in hops between D_1 and D_2 is either 2 or 3. Therefore, for any node u , if to u 's knowledge, all dominators that are at most two hops away from u are connected with each other, D will be connected. In other words, if u itself is a dominator, u needs to connect to all nodes in $N_2(u) \cap D$; otherwise, $\forall v \in N_1(u) \cap D$ should be connected to $\forall w \in N_2(u) \cap D$. Here $N_i(u)$ is the set of nodes that are exactly i -hop away from u for $i = 1, 2$.

Based on these observations, we design a localized procedure to connect D . Note that the necessary neighborhood information has been collected in Phase I. Let $List_1^u$ be the set of dominators that are *at most* one-hop away from u . Therefore if u is a dominator after Phase I, $u \in List_1^u$. Let $List_2^u$ be the set of dominators that are *exactly* two-hop away from u . Let S_u be the set containing all pairs (u_1, u_2) such that $u_1 \in List_1^u$, $u_2 \in List_2^u$, and $ID_{u_1} < ID_{u_2}$. Note that the two nodes of any pair in S_u are at most three hops away. Our objective in Phase II is to connect all pairs of dominators in S_u so that D is connected. If $S_u \neq \phi$, u may join the CDS as a connector and broadcast a connector message.

When the network density is high, the above mentioned procedure results in a CDS with very large cardinality. To overcome this problem, we introduce a *distributed competition* among neighboring nodes. Each node u maintains a timer $Timer_2^u$, which is initialized as $\delta_1 + Rand_u(\delta_1)$ if u is a dominator, and to $\delta_2 + F(u)$ if u is a *dominatee*. $F(u)$ is a customizable function, and will be explained further in the remarks. When $Timer_2^u$ times out, u wins the competition and broadcasts a *connector* message to invite other nodes to join the CDS in order to connect all pairs in S_u . A neighbor node v is to be invited only if there exist two dominators x and w such that $(x, w) \in S_u$, and u is connected to w via v (u has the information from Phase I). Note that for the same w , u only invites one neighbor node. Upon receiving a *connector* message before $Timer_2^u$ times out, u stops its timer. If the sender of the *connector* message is v , u updates S_u by removing all pairs of (x, y) in the S_v (Note that u can obtain S_v by checking $List_2^v$ in the *connector* message and $List_1^v$ from Phase I). If u is included in the $List_{invi}$ field of the received *connector* message, u joins the CDS as a connector and broadcasts a

connector message immediately². No matter u is invited or not, it needs to update S_u further by examining whether the joining CDS by v (and possible by u) can connect any pairs in S_u . If u is a *dominatee* with a non-empty S_u , but is not in the invitation list, it re-initializes and restarts its timer. From the above description, we notice that a dominator has a higher priority over the neighboring *dominatees* to invite nodes into the CDS. In addition, a dominator u always broadcasts a *connector* message if its $S_u \neq \phi$.

The following two cases summarize the above description:

- **Case I: The node u is a dominator.** If $S_u \neq \phi$, u broadcasts a *connector* message inviting just enough neighbor nodes to connect all pairs in S_u . Depending on the custom value of its neighbors, it is possible for u to prefer some nodes over others. Or, u can just randomly choose a subset of the neighbors as long as all pairs in S_u can be connected.
- **Case II: The node u is a *dominatee*.** If u wins the local competition and $S_u \neq \phi$, u becomes a CDS node and broadcasts a *connector* message inviting other necessary nodes to join the CDS such that all pairs in S_u can be connected. The *connector* message also serves another purpose of announcing u 's joining the CDS. Upon receiving a *connector* message during the competition, u stops its timer and updates S_u . If u is invited to join the CDS by the *connector* message, u also wins the competition. Otherwise, if $S_u \neq \phi$, u resets its timer to $\delta_2 + F(u)$ and participates in the competition as before.

The state transition of a node u in Phase II is illustrated in Fig. 4. The detailed description is given below.

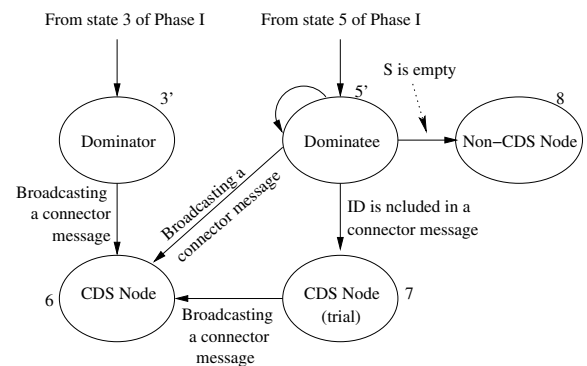


Figure 4. Node state transitions in Phase II.

- **3 → 3'**: The node u is a dominator at the end of Phase I. u enters Phase II if the media is clear for at least δ_3 time after receiving a *dominatee* message.

²It is possible for a *connector* message to have an empty $List_{invi}$.

- **5** → **5'**: The node u is a dominatee at the end of Phase I. u enters Phase II if the media is clear for at least δ_3 time after broadcasting a dominatee message.
- **State 3'**. If $S_u = \phi$, it enters state 6 directly; otherwise, after the media is clear for $\delta_1 + Rand_u(\delta_1)$ time, u broadcasts a connector message to connect all pairs in S_u , and then enters state 6.
 - **3' → 6**: u enters the final state 6 if u does not have any two-hop dominator neighbors, or after it broadcasts a connector message to connect to all of its two-hop dominator neighbors.
- **State 5'**. If $S_u \neq \phi$, $Timer_2^u$ is initialized to $\delta_2 + F(u)$.

Upon successfully timing out, u does the following:

- **5' → 6**: u broadcasts a connector message and joins the CDS.

Upon receiving a connector message, u updates S_u and conducts one the following state transitions:

- **5' → 5'**: $S_u \neq \phi$ and u is not in the $List_{invi}$ field of the received message.
- **5' → 7**: u is invited to join the CDS by the sender.

If $S_u = \phi$, $Timer_2^u$ is initialized to δ_3 . If the timer successfully expires, which means no connector messages have been broadcast in its neighborhood, u becomes a non-CDS node.

- **5' → 8**: As $S_u = \phi$, u turns to be a non-CDS node.

- **State 6**. The final state for CDS nodes.
- **State 7**. If $S_u \neq \phi$, u first broadcasts a connector message as soon as the media is clear, and then goes to state 6; Otherwise, u goes to state 6 immediately.
 - **7 → 6**: u is invited to join the CDS.
- **State 8**. The final state for non-CDS nodes.

Remarks:

- We exploit δ_1 and δ_2 to prioritize the message broadcastings of Phase II.
- The introduction of the three time intervals δ_1 , δ_2 , and δ_3 is motivated by the SIFS, PIFS, and DIFS in IEEE 802.11 [14]. In this study, we choose $\delta_2 = 2 \times \delta_1$ and $\delta_3 = 2 \times \delta_2$. δ_1 should be long enough for a round-trip transmission in an one-hop neighborhood.

- $F(u)$ (for node u) is a function of the custom value CV . For collision avoidance, we also need to add a small random number such that two nodes with the same custom value do not broadcast messages at the same time with very high probability.

3.2 Customization and A Customized Algorithm

The customizability of our algorithm mainly lies in the decision of node roles (as dominators/dominatees/connectors). If a node decides its role as a dominator or a connector, it joins the connected dominating set, and therefore may assume more responsibility (e.g. forwarding traffic) than those not in CDS. If we have the knowledge that certain nodes should be placed in CDS (e.g. nodes with more energy or better communication capability), we can assign higher priority to them for becoming dominators or connectors. On the contrary, if some nodes should be refrained from taking more responsibility and thus from joining the CDS, lower priority should be given to them.

As a test of the customizability, we provide a customized version of the general algorithm described above. The goal is to achieve smaller CDS sizes. The following customizations are introduced:

1. We assume that before Phase I, all nodes spend some time to learn the amount of immediate neighbors. In Phase I that follows, the nodes with more undefined neighbors have larger initial p and smaller timeout value for $Timer^1$. In other words, the nodes with more undefined neighbors have higher priority of becoming dominators. During the whole course of Phase I, the number of undefined neighbors are kept updated.
2. In Phase II, the nodes that have the larger number of different nodes in their S set have higher priority of sending a connector message. However, we still prioritize dominators over dominatees.
3. When inviting neighbors to join CDS, we mimic the greedy algorithm to build multipoint relay sets. For a node u , let v be a node such that (x, v) is in S_u . If there is only one node x that connect u to v , add x into the invitation list. After adding all such nodes, we prioritize the rest neighbors according to their coverage of uncovered second nodes in S_u (i.e. v just mentioned).

The performance of the customized algorithm is shown in the simulation section.

4 Performance Analysis

In this section, by providing a few theorems, we study the performance of the general algorithm proposed in the previous section. Due the space limit, the proof is omitted and available in the technical report.

Theorem 1 *All nodes at state 3 in Phase I form a maximal independent set.*

Theorem 2 *All nodes at state 6 in Phase II form a connected dominating set.*

Theorem 3 *The time complexity of our localized CDS construction procedure is $O(d+t)$, where d is the average node degree.*

Theorem 4 *The total number of messages each node broadcasts is at most 3.*

5 Simulations

To observe the average CDS size, we have run simulations for our general algorithm and the customized version as well as for the MPR and EMPR algorithms. For our general algorithm, we use the same initial p for all nodes, and the timeout values and invitation sets are all randomly decided based on the same parameters.

The simulations are conducted in an 100×100 area with n nodes randomly distributed on it, where n ranges from 20 to 100 with the step size 10 and from 100 to 1000 with the step size 100. We have tested two transmission ranges: 20 and 40. We did not test transmission ranges smaller than 20 since it is hard to obtain connected graphs for small number of nodes. For each setting, we run the simulation 100 times and calculate the average.

5.1 The Effect of p 's Initial Values

Before comparing the performance of the algorithms, we have checked with the effects of p 's initial value on the general algorithm. We tested six different values of p : $1/2$, $1/4$, $1/8$, $1/16$, $1/32$ and $1/64$. It turns out that the differences between these initial values of p do not change the performance of our algorithm (the general version) much, as shown in Fig. 5.

5.2 Comparison among Different Algorithms

Fig. 6 displays the average CDS sizes of all four algorithms for various settings. In the figures, the curves of the MPR algorithm, the EMPR algorithm, our general algorithm and the customized version are denoted as MPR,

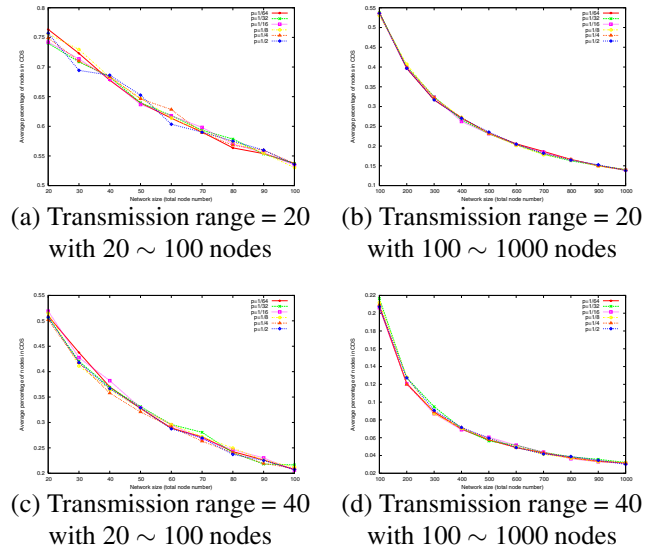


Figure 5. Insignificant impact of initial p on the average CDS size.

EMPR, CL and CL-e respectively. The displayed results of our general algorithm are from $p = 1/64$. We observe that the general version of our algorithm cannot achieve as small CDS on average as MPR and EMPR can. The average sizes almost double. It is understandable as the cost of customizability: in Phase I, some nodes are randomly chosen as dominators and added into the CDS, and in Phase II, the connections have to be made based on these nodes while the connections are made somehow randomly. If the choices are made more carefully, it is possible to gain better performance. In fact, the average CDS sizes of the customized version are quite close to those of MPR and EMPR. It indicates that the general algorithm has the potential to gain better performance after being customized.

6 Conclusions

We have proposed a customizable algorithm to construct connected dominating sets (CDS) distributively using only local (specifically, within two hops) information. The algorithm can be applied to self-organizing wireless networks to improve the performance of routing, broadcasting, topology control and energy conservation, etc.

The algorithm is customizable in that users can tune some parameters in the network nodes and prioritize them for joining the CDS. It is purely localized as its time complexity is $O(d+t)$ where d is the average node degree, and t is usually a small number (please refer to Section 3). The message complexity is also very low. Each node sends at most 3 messages during the construction of CDS.

However, the customizability comes at a price. Using

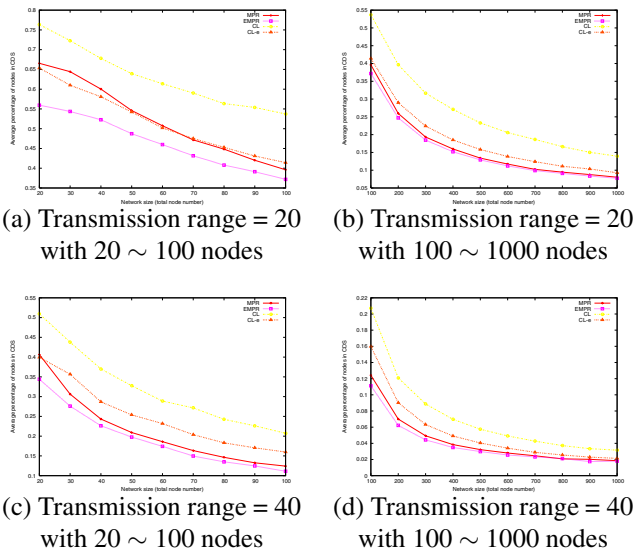


Figure 6. Average CDS sizes of all four algorithms for various settings.

simulations, we compare the average CDS sizes of MPR [8], EMPR [9] and our algorithm. The former two both outperform ours. On the other hand, our algorithm has the potential to obtain better performance with the right customization. We show that by giving a customized version of our general algorithm. In simulations, its average CDS sizes are close to those of MPR and EMPR. Of course, in many cases CDS size is not the only metric for performance evaluation. However, the actual metrics that should be considered depend on the applications, and thus vary from case to case. Therefore, we left out other metrics when checking performance, and only used CDS sizes as it is important in many aspects.

Future research is planned on CDS construction for dynamic network topology and CDS maintenance with the proposed algorithm as the basis. We will also consider the impact of customization on other performance metrics.

7 Acknowledgement

Dr. Jiang Li's research was partially supported by the National Science Foundation under grant No. 0324818.

References

[1] J. Wu, H. Li: On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks. *Proc. of the 3rd Int'l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications.* (1999) pp. 7-14.

[2] S. Datta, I. Stojmenovic: Internal Node and Shortcut Based Routing with Guaranteed Delivery in Wireless Networks. *Cluster Computing.* **5(2)** (2002) pp. 169-178.

[3] B. An, S. Papavassiliou: A mobility-Based Clustering Approach to Support Mobility Management and Multicast Routing in Mobile Ad-Hoc Wireless Networks. *International Journal of Network Management.* **11** (2001) pp. 387-395.

[4] J. Wu, F. Dai: Broadcasting in Ad Hoc Networks Based on Self-Pruning. *IEEE INFOCOM.* (2003).

[5] M. Ding, X. Cheng, G. Xue: Aggregation Tree Construction in Sensor Networks. *Proc. of IEEE VTC.* (2003).

[6] B. Deb, S. Bhatnagar, B. Nath: Multi-Resolution State Retrieval in Sensor Networks. *First IEEE Workshop on Sensor Network Protocols and Applications (SNPA).* (2003).

[7] M.R. Garey, D.S. Johnson: Computer and Intractability: A Guide to The Theory of NP-Completeness. *W. H. Freeman.* (1979)

[8] C. Adjih, P. Jacquet, L. Viennot: Computing Connected Dominated Sets with Multipoint Relays. *Journal of Ad Hoc and Sensor Wireless Networks.* Vol. 1. (2005).

[9] J. Wu : An Enhanced Approach to Determine A Small Forward Node Set Based on Multipoint Relays. *Proc. of 2003 IEEE Semiannual Vehicular Technology Conference (VTC2003-fall).* (2003).

[10] S. Guha, S. Khuller: Approximation Algorithms for Connected Dominating Sets. *Algorithmica.* **20(4)**. (1998). pp. 374-387.

[11] X. Cheng, M. Ding, D. Chen: An Approximation Algorithm for Connected Dominating Set in Ad Hoc Networks. *Proc. of International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peer-to-Peer Networks (TAWN).* (2004).

[12] B. Das, V. Bharghavan: Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets. *International Conference on Communications.* (1997).

[13] X. Cheng, M. Ding, D.H. Du, X. Jia: On The Construction of Connected Dominating Set in Ad Hoc Wireless Networks. *Special Issue on Ad Hoc Networks of Wireless Communications and Mobile Computing.* (2004).

[14] B.P. Crow, I. Widjaja, J. Geun Kim, P.T. Sakai: IEEE 802.11 wireless local area networks. *IEEE Communications Magazine.* **35.** (1997). pp. 116 - 126.