Iterative Local Solutions for Connected Dominating Sets in Ad Hoc Wireless Networks *

Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

Fei Dai
Microsoft Corporation
Redmond, WA 98052

Shuhui Yang
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180

*The work was supported in part by CCR 0329741, CNS 0422762, CNS 0434533, CNS 0531410, and CNS 0626240.
Corresponding author: jie@cse.fau.edu.

Abstract

In this paper, we propose a general framework of the *iterative local solution* (ILS) for computing a connected dominating set (CDS) in ad hoc wireless networks, which include mobile ad hoc networks (MANETs) and wireless sensor networks (WSNs). This approach uses an iterative application of a selected local solution. Each application of the local solution enhances the result obtained from the previous iteration, but each is based on a different node priority scheme. Then, we integrate this iterative process into the process for handling a dynamic network topology and propose two extensions: *cyclic iterative local solution* (CILS) and *seamless iterative local solution* (SILS). CILS offers a natural extension of ILS to the dynamic environment, but suffers from a broken CDS and non-adaptiveness. With a novel use of a monotonically increasing sequence number for dynamic node priority, SILS offers an extension with the desirable properties of correctness, progressiveness, locality, and seamlessness. Extensive simulations are conducted under *ns-2* and a custom simulator to evaluate the effectiveness of the proposed approach in both static and dynamic environments.

Keywords: *Connected dominating set (CDS), dynamic node priority, local solution, mobile ad hoc networks (MANETs), simulation, wireless sensor networks (WSNs).*

1 Introduction

In ad hoc wireless networks, which include mobile ad hoc networks (MANETs) and wireless sensor networks (WSNs), various algorithmic solutions can be classified into *global, quasi-global, quasi-local*, and *local* [36] depending on the amount of information used by each node to determine a solution for a specific problem (e.g., connected dominating set (CDS) as a virtual backbone in MANETS [35] and for coverage in WSNs [8], and network topology control for saving energy and reducing signal interference in MANETs [9]). The local approach uses local information to determine node status and such status does not propagate; i.e., the status of each node does not depend on the status of its neighbors. Therefore, the local approach is the most desirable to support scalable design through localized maintenance in a dynamic environment (also called locality). In the construction of a CDS, the status of a node is either inside or outside the selected CDS; whereas in network topology control, the status of each node is the selected transmission range for the node.

One potential problem of local solutions is relatively low efficiency (i.e., the quality of results). In CDS construction the quality is measured by the size of the CDS, and in topology control the quality

is quantified by the transmission range subject to connectivity. In this paper, we present a general framework of the *iterative local solution* (ILS) that relaxes the non-propagation constraint of local solutions in order to improve efficiency. Each application of a selected local solution enhances the result obtained from the previous iteration, but based on a different node priority scheme. However, ILS still keeps locality; that is, ILS can quickly provide a solution after a network topology change.

Figure 1 shows the difference between global, local, and iterative local solutions where time is slotted into rounds, each of which is a square block. Each round is measured as one or more "Hello" message exchanges in ad hoc wireless networks. To simplify the discussion, local solutions take one round to generate a solution. Both gray and black blocks correspond to correct results generated at respective rounds. In this image, the darker the color of the block is, the higher the efficiency of the result generated at the corresponding round will be. ILS quickly generates a result, albeit inefficiently, and then improves it over the iterations before the next network topology change (represented by a vertical line in the figure). In global solutions, an efficient solution can be generated after several rounds (say r). However, if the network topology changes frequently, no results can be generated in global solutions, as in Figure 1 where the distance between two changes (c_2 and c_3) is less than r. Note that in ILS, nodes exchange new node priority and node status between rounds, while in global solutions nodes exchange link state information.

In this paper, we focus on using ILS to calculate a CDS with the objective of reducing the CDS size over a number of iterations. Here ILS is first discussed in a static environment, followed by its extensions in a dynamic environment. This framework is illustrated using Dai and Wu's Rule K [14], an extension of Wu and Li's marking process [35], as a local solution. Each node determines its status: *marked* (inside CDS) or *unmarked* (outside CDS), based on local topology information and node priority in the neighborhood. Basically, a node can be unmarked if its neighborhood can be covered (dominated) by a set of nodes with higher priorities, and these nodes are connected by themselves.

Dominating sets (DS) have been widely used in the selection process of active node sets in ad hoc wireless networks. A set is dominating if every node in the network is either in the set or a neighbor of a node in the set. When active nodes form a dominating set, all nodes in the network are also said to be *reachable*. When a DS is connected, which means any two nodes in the DS can be connected through intermediate nodes from the DS, it is denoted as a connected dominating set (CDS). A CDS as a virtual backbone has been widely used in broadcasting [27] in MANETs and data aggregation [8]

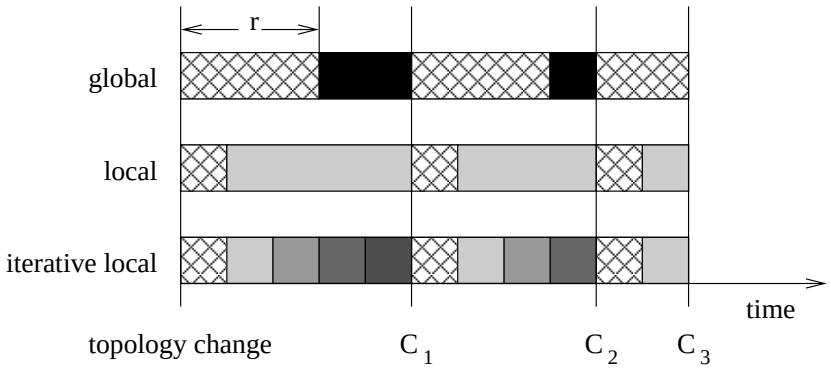


Figure 1: A comparison between global, local, and iterative local solutions.

in WSNs. In efficient broadcasting, only dominating nodes need to forward the broadcast packet, and all remaining nodes can receive the packet without having to forward it. In data aggregation, each dominating node collects data in its neighborhood, and then aggregates and forwards the data to the sink. CDS has also been used in the routing protocol OLSR [12], in which link state information is stored only in dominating nodes.

When applying ILS to a static environment, each node gathers topology information within h hops (for a small constant h), collecting h-hop information in what is called one round (iteration), and then determines its node status, marked or unmarked, through an iterative process with a constant number (k) of iterations. During each iteration, nodes are assigned different priorities so that more nodes can be unmarked as the process iterates. When applying ILS to a dynamic environment, the challenges lie in seamlessly blending topology changes into the scheme so that the following properties are maintained:

- **Correctness**: The CDS should be maintained at the end of each iteration (round) unless a new topology change occurs during the iteration.
- **Progressiveness**: The CDS size should be monotonically decreasing between iterations when there is no topology change.
- **Locality**: A topology change only affects the status of nodes in the local neighborhood, where the hop count in such a neighborhood depends on h.
- **Seamlessness**: The effects of the iterative process and topology change are integrated in a seamless way.

We propose two extensions of ILS: *cyclic iterative local solution* (CILS) and *seamless iterative local solution* (SILS). CILS offers a natural extension of ILS to the dynamic environment where ILS is cyclically applied for every k iterations. However, none of the above properties can be maintained. With a novel use of a monotonically increasing sequence number for dynamic node priority, SILS offers an extension with the desirable properties of correctness, progressiveness, locality, and seamlessness.

The major contributions of this paper are the following:

1. We devise a generic framework for the notion of iterative local solutions with a controlled iteration number.
2. We propose an iterative local solution in a dynamic environment that ensures correctness, progressiveness, locality, and seamlessness.
3. We offer novel use of a monotonically increasing sequence number as the dynamic node priority in a dynamic environment to seamlessly integrate the iterative process with topology changes.
4. We provide extensive simulation results under *ns-2* and a custom simulator for various trade-offs in different environment settings.
5. We point out other applications of ILS, including network topology control in MANETs and area coverage in WSNs.

The remainder of the paper is organized as follows: Section 2 discusses some related work, with a focus on iterative local solutions (ILS). Section 3 presents the general framework of iterative local solutions in a static environment. This framework is illustrated using Dai and Wu's Rule K as a sample local solution. The section concludes with a discussion on various ways of generating dynamic node priority based on node IDs. In Section 4, the drawbacks of CILS as a natural extension of ILS are discussed. Then we propose SILS and present its properties, followed by a special implementation. Several extensions of SILS are provided in Section 5. Section 6 offers extensive simulations on a custom simulator. Various trade-offs are shown in different settings in both static and dynamic environments. The paper concludes in Section 7 with a brief discussion of future directions.

2 Related Work

Our objective is to find a CDS that covers a unit disk graph representing a MANET based on local information. The problem of finding a minimum CDS (MCDS) is NP-complete for both general graphs [11] and unit disk graphs [26]. Wu and Lou [36] gave a comprehensive classification of heuristic CDS algorithms in MANETs: *global*, *quasi-global*, *quasi-local*, and *local*. Global solutions, such as Das, Sivakumar, and Bharghavan's MCDS [15] and Guha and Khuller's greedy algorithm [19], are based on global state information and are expensive in MANETs. Quasi-global solutions, such as tree-based CDS approaches [5, 33], require network wide coordination, which causes slow convergence in large scale networks. Many cluster-based approaches [4, 23, 36] are quasi-local. The status (clusterhead/non-clusterhead) of each node depends on the status of its neighbors, which in turn depends on the status of neighbors' neighbors and so on. The propagation of status information is relatively short ($O(\log n)$) on average, but can span the entire network in the worst case. Dubhashi et al [16] proposed another quasi-local approach, with a bounded ($O(\log n)$) steps of status propagation.

In local approaches (i.e., localized algorithms), the status of each node depends on its k-hop information only with a small k, and there is no propagation of status information. Local CDS formation algorithms include Wu and Li's marking process (MP) [35], several MP variations [10, 14, 31, 34], Sinha, Sivakumar, and Bhargharan's CEDAR [28, 29], Qayyum, Viennot, and Laouiti's multipoint relay (MPR) [27], and MPR extensions [1, 22, 25].

Many local solutions rely on node priorities to avoid simultaneous withdrawals in mutual coverage cases. One drawback of these priority-based schemes is that they may select a large CDS based on a bad priority assignment. Attempts have been made to mitigate this problem. For example, Stojmenovic [30, 31] proposed to reduce the CDS size via adaptive interpretation of priority values. In these schemes, the priority assignment is fixed; therefore, they cannot effectively eliminate redundant dominating nodes. In [3], a mechanism is applied to dynamically maintain the CDS property in a dynamic environment as opposed to dynamically reducing the CDS size over time.

Several iterative approaches have been proposed to find a small DS [18, 21] or CDS [24] in MANETs. In [18], Gao et al gave a basic algorithm to find a DS with an expected approximation ratio of $O(\sqrt{n})$, where each node designates a node with the highest priority in its neighborhood as a dominator. To obtain an expected $O(1)$ approximation ratio, the basic algorithm is repeated $\log(\log n)$ times using exponentially growing transmission ranges. In another iterative DS algorithm proposed

by Kuhn and Wattenhofer [21], each node v becomes a dominator with a probability p_v. If there are still uncovered nodes (i.e., nodes without neighboring dominators) after this process, these uncovered nodes also become dominators. The probability p_v is computed via a distributed linear programming algorithm that takes k^2 iterations with an adjustable parameter k. The iterative algorithm has an expected approximation ratio of $O(k\Delta^{2/k} \log \Delta)$, where Δ is the maximal node degree.

Liu, Pan, and Cao [24] proposed an iterative extension of Wu and Li's marking process and Rules 1 and 2 [35] for the local construction of a CDS. In the marking process, a node becomes a dominator (*marked*) if it has two neighbors that are not directly connected. According to Rule 1, a marked node can change back to a non-dominator (unmarked), if all its neighbors are also neighbors of another marked node with a higher priority (called a *coverage node*). In Rule 2, a marked node can be unmarked if its neighbor set is covered jointly by two connected coverage nodes. The iterative extension takes six rounds. The marking process is applied in round 1. Rule 1 is applied in round 2 with one priority (lower node ID has a higher priority) and round 3 with another one (higher node ID has a higher priority). Finally, Rule 2 is applied in rounds 4, 5, and 6 with different priority functions. This approach produces a smaller CDS than the original marking process and Rules 1 and 2.

None of the above approaches address the CDS maintenance issue in dynamic networks where topology changes, such as link switched-on/off and node switched-on/off, occur during the iterative process. This paper proposes an iterative scheme that integrates the CDS maintenance mechanism into the iterative CDS reduction process, and maintains a CDS at each round of iteration.

3 Iterative Local Solution (ILS) in a Static Environment

This section starts with a general model for iterative local solution (ILS), which extends a scheme proposed by Liu, Pan, and Cao [24]. Dai and Wu's Rule K [14] is used as an example to illustrate the model. The section ends with a discussion of various ways of generating dynamic node priority based on node IDs.

3.1 General model

Algorithm 1 shows a k-round ILS, where local topology information can be defined in different

Algorithm 1 k-round Iterative Local Solution (at each node v)

1: Each node collects local topology information and applies a local solution to determine its status (marked or unmarked).
2: The process completes if the number of iterations reaches k; otherwise, each node selects a new priority and exchanges status (and priority if needed) with neighbors.
3: Apply the local solution again based on new node status and node priority. Go to step 2 for the next iteration.

ways. One possible definition is the h-hop information that will be discussed in the next subsection. Again, we assume that the collection of h-hop information corresponds to one round. The number of iterations k is a constant and adjustable parameter. The next two subsections focus on the selection of a local solution and node priority.

3.2 Local solution selection

We use an extension of Wu and Li's marking process [35], called Rule K [14], in the iterative local solution (ILS). The following rule is among the most efficient (in terms of producing a small CDS) non-iterative local solutions.

Algorithm 2 Rule K as Local Solution

A node is unmarked if its neighbors form a clique, or are dominated by a set of connected nodes with higher priorities.

Formally speaking, a MANET can be represented by an undirected graph $G = (V, E)$, where V is the set of nodes and E the set of links. $N(v) = \{u|(u, v) \in E\}$ denotes the neighbor set of v. Given a node set $S \subseteq V$, $N(S) = \bigcup_{v \in S} .N(v)$ is the set of nodes dominated by S. A node v can be unmarked if

1. $(u, w) \in E$ for all $u, w \in N(v)$, or
2. there exists a set of *coverage nodes* $S = \{v_1, v_2, ..., v_K\}$, such that v_1, v_2, ..., v_K have higher priorities than v, the derived subgraph $G(S)$ is connected, and $N(v) - S \subseteq N(S)$.

Applying Rule K requires *h-hop information* for $h \geq 2$. By h-hop information we mean the topology and other relevant information (e.g., node priorities) collected at each node via h "Hello"

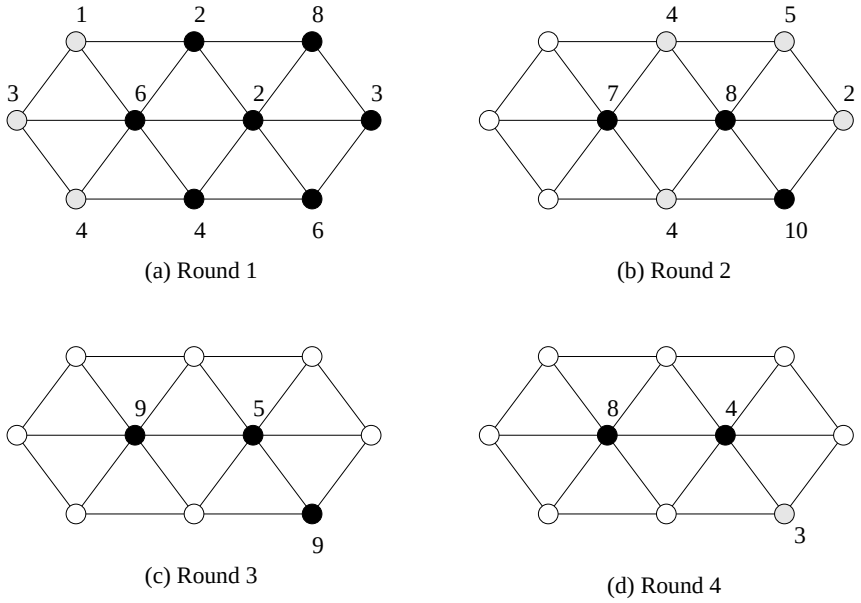
message exchanges among neighbors. For each node v, its h-hop information is a subgraph $G_h(v) = (N_h(v), E_h(v))$ of the MANET. $N_h(v)$ is v's h-hop neighbor set, defined as follows: $N_0(v) = \{v\}$ and $N_h(v) = \bigcup_{u \in N(v)} N_{h-1}(u)$ for $h \geq 1$. $E_h(v)$ are links among h-hop neighbors, excluding links between two nodes that are exactly h hops away from v; that is, $E_h(v) \subseteq (N_{h-1}(v) \times N_h(v))$. The overhead for collecting h-hop information is h messages per node. Each message includes $G_{h-1}(v)$ of the current node v and is of size $O(\Delta^{h-1})$, where Δ is the maximal node degree. A small h should be used to balance performance and overhead, such as $h = 2$ in the restricted Rule K, which incurs $O(\Delta)$ messaging cost and $O(\Delta^2)$ computing cost per node [14].

In order to use Rule K in the iterative local solution, h-hop information should also include the priorities and status of h-hop neighbors. In addition, the following restrictions are observed:

1. Initially, all nodes are considered marked.
2. At each round, only marked nodes use Rule K to determine their status, marked or unmarked, after this round. Unmarked nodes stay unmarked.
3. When applying Rule K, only marked nodes can be used as coverage nodes to unmark other marked nodes.

The resultant iterative local solution is called the *iterative Rule K*. Figure 2 shows a sample execution of iterative Rule K on a static network with 10 nodes. The restricted Rule K is used, i.e., $h = 2$. Each node is assigned a random priority at each round (iteration), which is visible to its neighbors. In round 1, three nodes with priorities 1, 3, and 4 are unmarked (represented by gray circles), because their neighbors are also neighbors of a node with a priority 6. Other nodes are marked (represented by black circles), which form a CDS. In round 2, the status of three unmarked nodes (represented by white circles) is propagated to their neighbors, which will not consider them as coverage nodes in applying Rule K. According to the new priority assignments, four nodes with priorities 4, 4, 5, and 2 are unmarked. In round 3, no node is unmarked based on the new priority assignment. In round 4, however, another node with priority 3 is unmarked.

Let $V_1, V_2, \ldots, V_k$ denote the sets of marked nodes after iterations $1, 2, \ldots, k$, respectively. The following theorem shows the correctness and effectiveness of the iterative Rule K. Here we assume a static network that is connected but not completely connected.



(a) Round 1 (b) Round 2

(c) Round 3 (d) Round 4

Figure 2: The first four iterations (a-d) of iterative Rule K in a static network with 2-hop information (i.e., Rule K is restricted). Black nodes are marked (i.e., in the CDS), white nodes are unmarked, and gray nodes are newly unmarked at each round. The labels of the black and gray nodes denote their priorities. Priorities of white nodes are irrelevant and omitted.

Theorem 1 *In iterative Rule K, V_i is a CDS for all $1 \leq i \leq k$, and $|V_{i+1}| \leq |V_i|$ for all $1 \leq i \leq k-1$.*

Proof: It has been proven in [14] that Rule K with the above constraints preserves the CDS property. That is, if a node set V_i is a CDS of a network G, then after applying Rule K with the above restriction 3, the resultant marked node set V_{i+1} is still a CDS of G. Let V_0 be the set of all nodes in the network. Obviously, V_0 is a trivial CDS of G. Therefore, V_1, the resultant marked node set of applying Rule K on V_0, is also a CDS of G. Similarly, $V_2, V_3, \ldots, V_k$ are all CDSs of G.

The remaining part of the theorem, $|V_{i+1}| \leq |V_i|$ is implied by restriction 2. Since an unmarked node will not become a marked node during the k iterations, the number of marked nodes will never increase. □

After enough rounds of iteration, the marked node set is stabilized; that is, no more marked nodes can be unmarked regardless of the priority assignment.

Definition 1 *An iterative local solution is stabilized at round k' if the set of marked nodes does not*

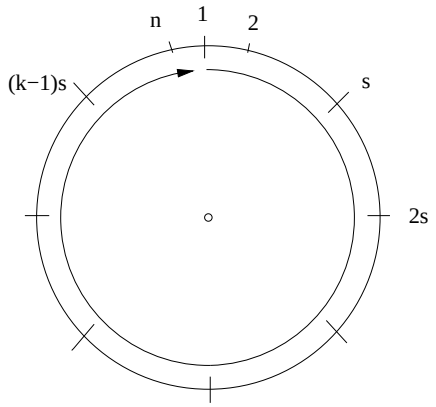


Figure 3: Priority rotation through circular s-shift.

change after round k', i.e., $V_{k'} = V_{k'+1} = \ldots = V_k$.

In the sample network of Figure 2, the iterative Rule K is stabilized at round 4. The two marked nodes in Figure 2 (d) cannot be unmarked based on any priority assignment. When the iterative Rule K takes k' rounds to stabilize, the sets of marked nodes $V_1, V_2, \ldots, V_{k'}$ change significantly in terms of both set size and set members. The specific value of k' depends on the priority rotation scheme and network topology. For example, the number of marked nodes in Figure 2 is 7, 3, 3, and 2 in the first four rounds. A good priority assignment scheme should achieve a fast convergence, i.e., stabilized in round k' with a small k', and converge to a small CDS as well.

3.3 Node priority rotation

There are several ways to rotate node priority (the corresponding scheme is called dynamic node priority). Here we denote priority as a function $p(v, i)$ of round number i and node ID v, where the IP (or MAC) address of each node can be used as its ID. To simplify the discussion, we assume that the initial priorities of n nodes are integers taken from $[1...n]$. In reality, a hash function can be used to map an IP address to an integer priority in $[1...n]$. Different nodes can have the same hash value as priority, since many local solutions (including Rule K [14]) support the same node priority case, but with less efficiency. In fact, as long as no conflict exists in the local neighborhood, efficiency will not be sacrificed.

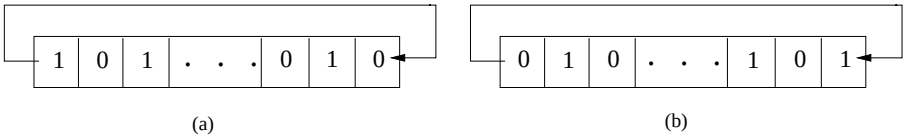


Figure 4: Perfect shuffle: (a) binary string as node ID, (b) binary string after one circular left shift.

Preposition 1 *If* $n = (\Delta^h)^2$ *is a hash function randomly chosen from a universal class of hash functions, then the probability of a node priority collision with any node in its* h*-hop neighborhood is less than* $1/(2\Delta^h)$.

This preposition can be easily derived from a result in [13]. Note that when h is small the condition $n = (\Delta^h)^2$ can be easily satisfied.

In the following, we will examine three possible priority rotation schemes.

- **Shifting**. Initially, the priorities $p(v, 0)$ of all nodes $v \in V$ are $1, 2, \ldots, n$, respectively. At each round i, the priority of each node v is defined as

  $$p(v, i) = (p(v, i-1) + s) \bmod n$$

  where $s = \lfloor n/k \rfloor$. The pattern of the priority change can be described as a circular s-shift, which is shown in Figure 3.

- **Shuffling**. In this scheme, node priority is changed more dramatically from round to round following the *perfect shuffle* [32] scheme. Node priority, represented as a binary string, is circularly shifted left one bit per iteration (see Figure 4). That is,

  $$p(v, i) = (p(v, i-1) \times 2) \bmod 2^k + \lfloor p(v, i-1)/2^{k-1} \rfloor$$

  Here we assume the iteration limit k satisfies $2^k \geq n$.

- **Random**. Node priority is randomly selected from $[1..n]$ at each round, i.e.,

  $$p(v, i) = rnd() \bmod n$$

  where $rnd()$ is a random number generator. Several nodes can have the same priority. The major difference between the deterministic approach (including the above shifting and shuffling schemes) and the random approach is that in the former, neighbors exchange node status

(marked/unmarked) only (except in the first round, where initial node priorities are also exchanged), whereas in the latter, neighbors need to exchange both node status and random node priorities generated at the current round.

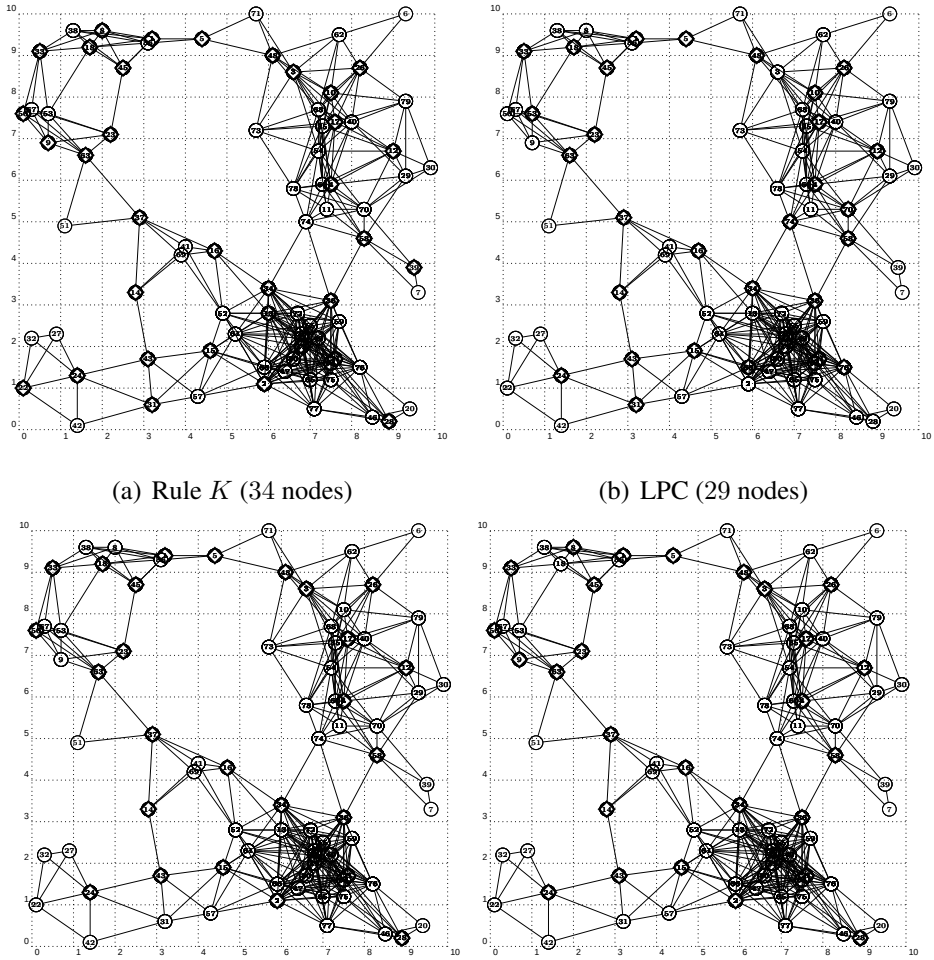
Figure 5 is a sample random network with 80 nodes. Rule K, ILS with different node priority rotation schemes, and the algorithm of Liu, Pan, and Cao [24] (denoted as LPC) are applied to generate the CDS. We can see that ILS (after 8 rounds) has smaller resultant CDS than LPC and Rule K. Among the three node priority rotations, *Random* has the best performance, and *Shuffle* has the same result as *Shifting*. The detailed performance of these protocols will be evaluated in the simulation section.

4 Seamless Iterative Local Solution (SILS) in a Dynamic Environment

In this section, we start with a natural extension, called cyclic iterative local solution (CILS), to be used in a dynamic environment. After pointing out several drawbacks of CILS, we give a novel extension of the iterative local solution, called the *seamless iterative local solution* (SILS). This solution is first presented as a generic scheme that specifies the critical requirements and desirable properties. A special case will then be discussed as a practical solution. The section ends with various relaxations to generalize SILS.

4.1 Cyclic iterative local solution (CILS)

In a static environment without topology changes, the iterative local solution can produce a small CDS after k' rounds of iteration, where k' is the number of rounds needed for stabilization. In a dynamic environment with node mobility (modeled as link switched-on/off operations and node switched-on/off operations), each node must reconsider its status periodically in order to maintain the CDS property. A natural, but somewhat nave, extension of the iterative local solution, called *cyclic iterative local solution* (CILS), can be used to handle topology changes. In this scheme, all nodes will reset their status and the process will start over again for every k rounds of iteration. That is, all nodes are considered marked again in round $k + 1$, and become gradually unmarked in the following k rounds:



(a) Rule K (34 nodes) (b) LPC (29 nodes)

(c) ILS & Random (26 nodes) (d) ILS & Shuffle/Shifting (27 nodes)

Figure 5: Sample CDS of different algorithms in a random network with 80 nodes.

$k+1, k+2, \ldots, 2k$. The same process will repeat in rounds $2k+1, 2k+2, \ldots, 3k$ and so on. Figure 6 (a) shows the general pattern of the CDS size with respect to the number of rounds. Such a scheme has certain limitations and we again use Rule K to illustrate.

However, CILS suffers from the following drawbacks:

- **Broken CDS**. The cyclic scheme guarantees a CDS, V_i, for $1 \leq i \leq k$ only if there is no topology change during these k iterations. If a topology change occurs in round i, then $V_{i+1}, V_{i+2}, \ldots, V_k$ may not be a CDS. For example, if the left node with priority 6 in Figure 2 (a) switches off after round 1, the set of marked nodes in the following rounds cannot form a CDS.

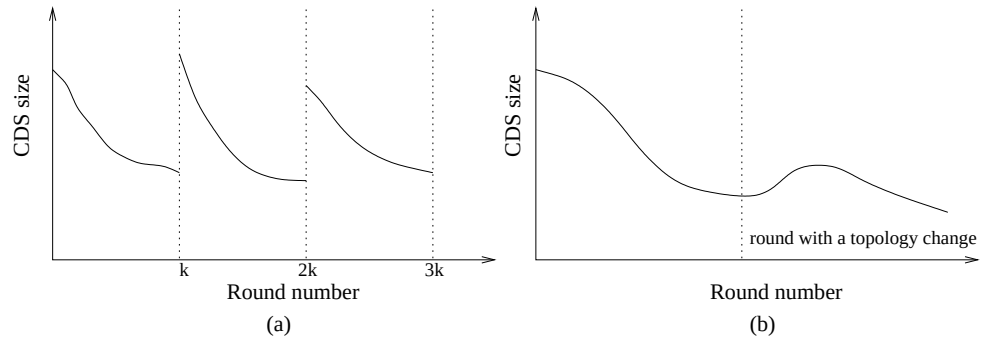


Figure 6: Two extensions to dynamic environments: (a) cyclic iterative local solution, (b) seamless iterative local solution.

- **Non-adaptiveness**. The selection of k in the cyclic scheme is non-adaptive. The cycle repeats even in a static environment. On the other hand, a large k will increase the probability of a broken CDS in a dynamic environment.

In fact, a broken CDS violates the correctness property. Non-adaptiveness destroys the progressiveness property, in which the CDS should be monotonically decreasing when there is no topology change. Non-adaptiveness also causes the seamlessness property to fail since an explicit counter is needed to keep track of iteration. The nature of cyclic application also breaks the locality property because the locality property must ensure that the number and selection of marked nodes does not change significantly after each topology change. The seamless iterative local solution (SILS) discussed in the next section meets all the above properties.

4.2 Seamless iterative local solution (SILS)

Again, we use Rule K to illustrate SILS, and the corresponding approach is called the *seamless iterative Rule K*. The basic idea is that the CDS formation process continues beyond k rounds of iteration. Node status (marked/unmarked) is adjusted in reaction to topology changes as the process iterates. These adjustments are conducted smoothly in small vicinities of topology changes without a global reset operation. Two important changes are made in this extension.

1. At each round, Rule K is applied at all nodes, marked or unmarked previously, to determine

their new status. Note that in the original iterative solution (Section 3.2), only a marked node may change its status as stated in restriction 2.

2. Node status is no longer exchanged among neighbors. The restriction 3 in the iterative Rule K (only marked nodes can be coverage nodes) is removed.

In the new extension, the original Rule K is applied based on h-hop information with a small h, including topology and priority information. At the beginning of each round i, each node collects the latest h-hop information through h rounds of "Hello" exchanges among neighbors. Each node v also selects its priority $P(v, i)$[1], which is embedded in the "Hello" messages and disseminated to its h-hop neighbors. The priority of a node can be any value that satisfies the following conditions:

1. $P(v, i) \geq P(v, i-1)$ for all $i \geq 1$.
2. $P(v, i) = P(v, i-1)$, if v is unmarked in round i.

The following theorem shows that the seamless iterative Rule K "repairs" a CDS in one round. If a topology change occurs in round $i-1$ and damages the CDS, a new CDS will be formed in round i, if no more topology changes occur in round i. This "repair rate" is the same as in the original (non-iterative) Rule K. Note that if the network topology changes in every round, no traditional local solution can maintain a CDS. Again, we use V_i to denote the set of marked nodes in round i, and assume that the network is connected in each round.

Theorem 2 (Correctness) *With the seamless iterative Rule K, V_i is a CDS in round i if there is no topology change in the current round.*

Proof: Let G_i be the network topology at the beginning of round i. As proved in [14], the marked node set V_i selected via Rule K is a CDS of G_i. Since no topology change occurs in this round, V_i is also a CDS within the entire duration of round i. □

The next theorem shows that the seamless iterative Rule K is as effective as the iterative Rule K in a static environment.

[1]Here we use a upper case P to distinguish the monotonically increasing priority in SILS from the one in CILS.

Theorem 3 (Progressiveness) *With the seamless iterative Rule K,* $|V_i| \geq |V_{i+1}| \geq \ldots \geq |V_j|$ *if there is no topology change in rounds* $i, i+1, \ldots, j$.

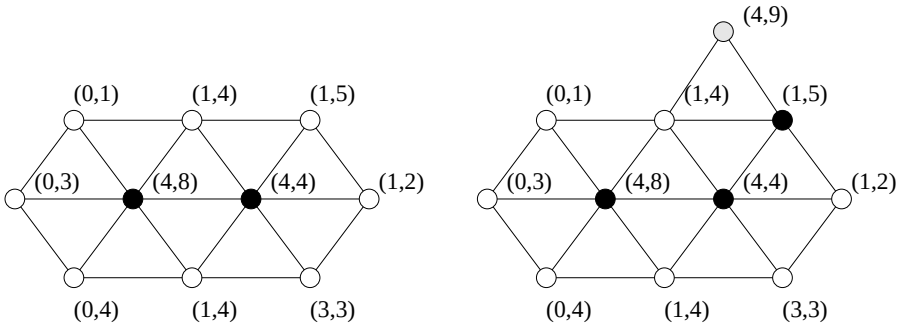
Proof: Suppose a node v is unmarked in round x ($i \leq x < j$). We show that v will stay unmarked in round $x+1$. Based on Rule K, if v is unmarked in round x, then all neighbors of v either (1) form a clique, or (2) are dominated by some connected coverage nodes $u_1, u_2, \ldots, u_K$ where $P(v,x) < \min\{P(u_1,x), P(u_2,x), \ldots, P(u_K,x)\}$. In either case, v will be unmarked in round $x+1$ if v's h-hop topology is not changed. In case (1), v's neighbors are still pairwise connected. In case (2), v's neighbors are still covered by nodes $u_1, u_2, \ldots, u_K$, and $P(v,x+1) = P(v,x) < \min\{P(u_1,x), P(u_2,x), \ldots, P(u_K,x)\} \leq \min\{P(u_1,x+1), P(u_2,x+1), \ldots, P(u_K,x+1)\}$. Therefore, $V_{x+1} \subseteq V_x$ for all $1 \leq x < j$. □

Finally we show that the effect of a topology change is localized. Specifically, when the seamless iterative Rule K uses h-hop information to determine the status of each node, the influence of a topology change is within $2h$ hops. We say a node v is within h hops of a topology change if such a change can be detected by v via collecting h-hop information.

Theorem 4 (Locality) *If the seamless iterative Rule K is stabilized at round* i*, then only nodes within* $2h$ *hops of a topology change may change their status after round* i.

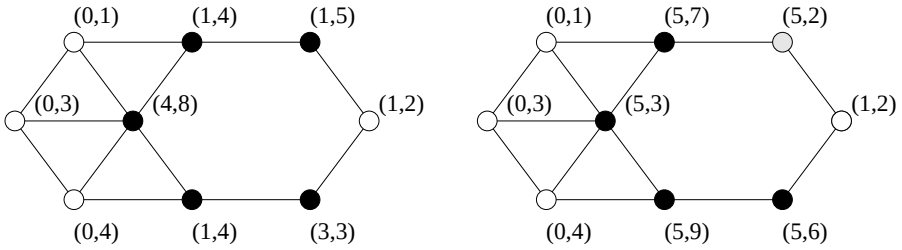
Proof: Let v be a node that changes its status in round $j > i$. We first consider the case that v is unmarked in round $j-1$ and marked in round j. From the proof of Theorem 3, v will stay unmarked as long as its h-hop topology is unchanged. Therefore, v is within h hops of a topology change.

Then we consider the case that v is marked in round $j-1$ and unmarked in round j. The theorem holds when v is within h hops of a topology change; otherwise, v's h-hop topology is unchanged. The only reason for v to be unmarked is priority changes in v's h-hop neighbors. Note that v's neighbor set cannot be dominated by a set of marked nodes in round i; otherwise, v will be unmarked in round $j > i$ when there is no topology change, which contradicts the assumption that the marking process is stabilized at round i. In order to cover $N(v)$, at least one node $u \in N_h(v)$ must have raised its status from unmarked to marked (and raised its priority) after round i. From the first part of this proof, u is within h hops of a topology change. Since v is at most h hops away from u, it is within $2h$ hops of this topology change. □



(a) Round 5, no topology change

(b) Round 5, node switch−on



(c) Rounds 5 and 6, node switch−off

Figure 7: A seamless iterative Rule K with 3-hop information. (a) Priority assignment after the first four rounds of iteration, where the random priority of each node in each round is the same as in Figure 2. (b) Handling a node switch-on event in round 5. (c) Handling a node switch-off event in rounds 5 and 6.

Note that the dynamic priority P integrates the treatment of both the iterative process and topology change in a seamless way. Figure 6 (b) shows a general pattern of CDS size. Notice that the CDS can increase as a response to a topology change (such as a newly switched-on node).

4.3 A special case

This subsection presents a special case of the seamless iterative Rule K, which is equivalent to the iterative Rule K in static networks and has all the desirable properties of the seamless iterative Rule K in a dynamic environment. For each round $i \geq 1$, the new priority of a node v is a 2-tuple (s, p), where s is a *sequence number*, which records the most recent iteration that a node was marked. A node with a higher sequence number has a higher priority. p is a secondary priority to break a tie between two nodes with the same sequence number. Let $p(v, i)$ be one of the priority rotation schemes

in Section 3.1.

$$P(v,i) = \begin{cases} (i-1, p(v,i)) & : \quad v \text{ is marked in round } i-1 \\ P(v,i-1) & : \quad \text{otherwise} \end{cases} \tag{1}$$

All nodes are considered *marked* in the first round. Therefore, when $i = 1$, the corresponding priority of each node v is $P(v,1) = (0, p(v,0))$. Any new node (e.g, a node that switches on in the current round) is also considered marked. A node v added at the beginning of round i has the priority $(i-1, p(v,i))$.

Figure 7 illustrates the seamless iterative Rule K using the priority function (1). In a static network (as shown Figure 7 (a)), the marking process is stabilized after round 4. At each iteration, the dynamic Rule K produces the same set of marked nodes as in the iterative Rule K (as shown in Figure 2). Note that nodes with priority $(i-1, p)$ are unmarked after iteration i. Figure 7 (b) shows a dynamic network where a new node is added (switches on) right before round 5. After detecting this topology change, a node with priority (1,5) is marked in round 5 to maintain a CDS, while the new node with priority (4,9) is unmarked immediately. Figure 7 (c) shows the situation of node switch-off. After the topology change is detected in round 5, four unmarked nodes become marked to form a CDS. In round 6, all marked nodes adjust their priority values. A newly marked node with the lowest priority is unmarked, producing a smaller CDS.

Theorem 5 *In a static network, the seamless iterative Rule K using the priority function (1) produces the same set of marked nodes as the iterative Rule K at each iteration.*

Proof: Let $P(v,i)$ and V_i denote the priority of node v and the set of marked nodes, respectively, in the iterative Rule K in round i. Let $P^{'}(v,i)$ and $V_i^{'}$ be the priority function and marked node set in the seamless iterative Rule K. In round 1 $P(v,i) = p(v,1)$. From equation (1), $P^{'}(v,i) = (0, p(v,1))$ because all nodes are considered marked before round 1. Obviously, $P(v,i) < P(u,i)$ iff $P^{'}(v,i) < P^{'}(u,i)$. Therefore, a node that is unmarked in the iterative Rule K is also unmarked in the seamless iterative Rule K; that is, $V_1 = V_1^{'}$.

Given $i \geq 1$ and $V_i = V_i^{'}$, we show that $V_{i+1} = V_{i+1}^{'}$. Suppose a node v is marked in round i and unmarked in round $i+1$ in the iterative Rule K. Node v is covered by several connected marked nodes $u_1, u_2, \ldots, u_K$ with $p(v,i+1) < \min\{p(u_1,i+1), p(u_2,i+1), \ldots, p(u_K,i+1)\}$. In the seamless

iterative Rule K, the priority of a marked node v in round i is $P'(v, i+1) = (i, p(v, i+1))$. Therefore, $P'(v, i+1) < \min\{P'(u_1, i+1), P'(u_2, i+1), \ldots, P'(u_K, i+1)\}$, and v is also unmarked in the seamless iterative Rule K. Similarly, any node unmarked in the seamless iterative Rule k will also be unmarked in the same round of the iterative Rule K. □

5 Extensions

In previous discussions, some simplified assumptions are used in SILS to ease discussion. This section shows that those assumptions can be relaxed, while preserving various desirable properties.

5.1 "Hello" message frequency

In Section 4.2, it is assumed that each node collects the latest h-hop information in each round. Therefore, each node will send h "Hello" messages per iteration. This requirement can be relaxed with the penalty of a slightly slower CDS repairing speed. Based on the relaxed requirement, each node sends only one "Hello" message per iteration. After a topology change occurs, it will be detected by all nodes within h hops in h rounds. Therefore, the CDS repair takes at most h rounds. Specifically, V_i is guaranteed to be a CDS if there is no topology change in rounds $i-h+1, i-h, \ldots, i-1, i$. Since all nodes may update their priorities at each round, the priority of a node in the h-hop information may be lower than its actual priority. This may cause conservative decisions (i.e., produce more marked nodes), but will not affect the correctness. All the localized algorithms have the same message complexity: $O(1)$ message per node, with message size of $O(\Delta)$, where Δ is the maximum node degree.

5.2 Asynchronous "Hello" messages

All previous discussions use a synchronous model. That is, the marking process is conducted in rounds. All nodes finish their work in the last round, including exchanging "Hello" messages and determining their status, before any node starts the next round. In fact, the SILS is still correct with asynchronous "Hello" messages and decision making processes. Suppose each node v has its unique "Hello" interval and re-computes its status whenever a new "Hello" message is received. CDS is still

guaranteed if there is no topology change in a recent time period of $h \times \max\{T_h(v)\}$, where $\{T_h(v)\}$ is a set of "Hello" intervals of v's h-hop neighbors.

One problem is determining the round number i in equation (1) at each node when there is no global synchronous iteration or round. The solution is to use a round number that is larger than all known sequence numbers. Specifically, the current round number with respect to a node v is $i = \max\{I_h(v)\} + 1$, where $\{I_h(v)\}$ is a set of sequence numbers belonging to v's h-hop neighbors.

5.3 Sequence numbers for new nodes

In Section 4.3, it is assumed that all new (switched-on) nodes have a sequence number that is equal to the current round number. Nevertheless, the special case is also correct if a smaller sequence number is selected. For example, if the new node in Figure 7 (b) uses a sequence number 0, which changes its priority to $(0, 9)$, the resultant set of marked nodes is still a CDS. Intuitively, using a large sequence number for new nodes can help unmark previously marked nodes. Using a small sequence number for a new node can help the new node unmark itself.

5.4 Sequence number recycle

In the previous discussion, we assume that the sequence number can increase infinitely without causing overflow. In a practical implementation, each sequence number occupies a finite number of bits, and is reset periodically. Reset occurs when the largest sequence number exceeds the maximum value, when all nodes will set their sequence numbers to zero. In synchronous systems, a reset operation can be triggered when the current round number reaches a given threshold. In asynchronous systems without a global round number, this operation can be initialized via a network wide broadcasting. The reset operation preserves correctness, but violates the progressiveness, locality, and seamlessness principles, as in CILS. The difference is that in SILS, the reset operations are much less frequent.

5.5 Simultaneous topology changes

The correctness of SILS depends on the accuracy of h-hop topology $G_h(v)$ collected by each node v. If there are no topology changes during a time period that is sufficiently large to complete h "Hello"

message exchanges, each node will have accurate h-hop topology information to select a CDS. Neither the type of topology changes (node or link switched-on/off) nor the combination of topology changes (one per round, several per round, or in the middle of two rounds) will affect the correctness of the algorithm. Therefore, there is no need to enforce "atomicity" of each change or to "serialize" these changes.

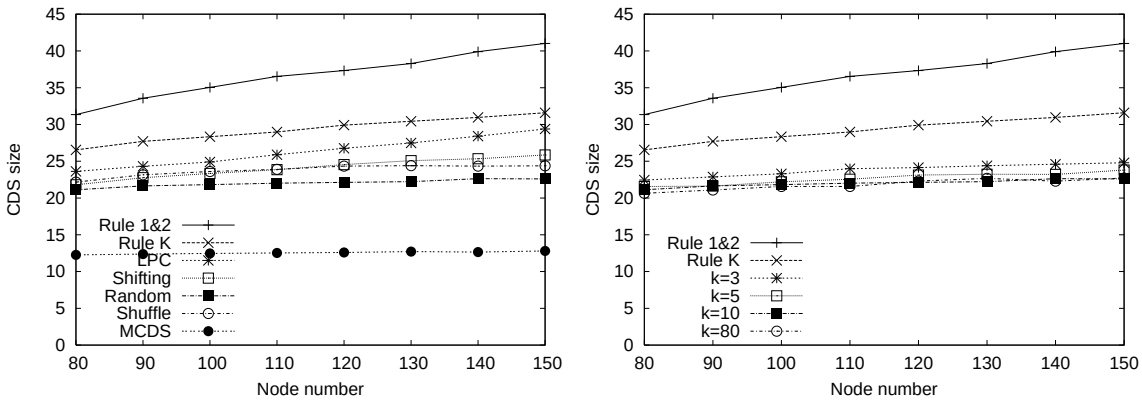
5.6 Time-driven vs. event-driven update

In above discussion, we assume a time-driven approach. All nodes periodically exchange messages and update their statuses using a fixed "Hello" interval. In semi-static networks with scarce topology changes, an event-driven scheme can be used to reduce the message overhead. The basic idea is that a node will stop sending "Hello" messages or update its status, if there is no topology change in its h-hop neighborhood.

Two difficulties exist in implementing the event-driven scheme. First, stopping the iterative process too soon may cause a large CDS size. As a solution, a node will do k more iterations after its neighborhood topology becomes static, where k is determined based on experimental data to balance performance and overhead. Second, each node relies on the "Hello" messages of its neighbors to detect topology changes. Totally eliminating "Hello" messages also disables this neighbor discovery mechanism. Instead, an adaptive scheme should be used: each node increases its "Hello" interval when no topology change happens, and decreases otherwise. The new "Hello" interval is advertised in the previous "Hello" message for the ease of link failure detection.

5.7 Selection of local solution

In this paper, we use Rule K as a sample local solution in ILS. Other local solutions that use node priority to prevent simultaneous withdrawals can use the same iterative approach to reduce the resultant CDS size. For example, both Wu and Dai's coverage condition [34] and Adjih, Jacquet, and Viennot's extended MPR [2] can be used in place of Rule K in ILS.



(a) Comparison of CDS size ($k = 8$)

(b) Effect of round number

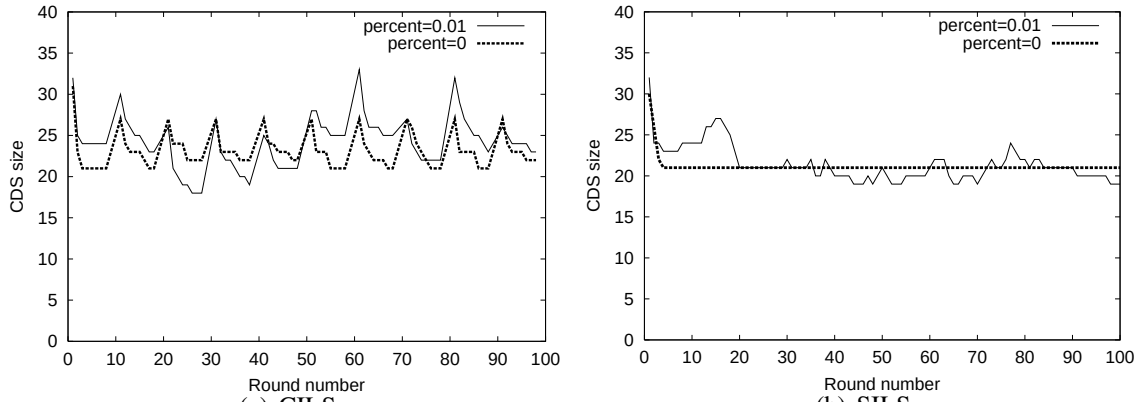
Figure 8: Performance of ILS in a static environment.

6 Simulation

Our simulation study has focused on the performance improvement after applying iterative local solution (ILS) on restricted Rule K, which has demonstrated high efficiency and reliability in dynamic networks in a previous simulation study [6]. It contains two parts. Part one focuses on the efficiency (in terms of the derived CDS size) of the iterative Rule K compared with existing CDS algorithms. This part is conducted via a custom simulator, which simulates a static network for the sake of simulation speed. Part two explores both efficiency and reliability (in terms of coverage), which requires a more realistic environment with topology changes and packet losses. This part is implemented on ns-2(1b7a) [17]. Two dynamic networks are considered: WSNs, where sensor node switch operations dominate, and MANETs, where topology changes are mainly caused by node movement.

6.1 Static environment

In this subsection, the performance of iterative Rule K is compared with Wu and Li's Rule $1\&2$ [35], Dai and Wu's Rule K [14], and the algorithm of Liu, Pan, and Cao [24] (denoted as LPC), using the custom simulator. The MCDS algorithm of Das et al [15] is a global CDS approach, which "grows" a tree from a selected root until all nodes are covered. Non-leaf nodes form a CDS. MCDS has an $O(\log \Delta)$ approximation ratio in regular graphs, where Δ is the maximum degree. The MCDS



(a) CILS (b) SILS

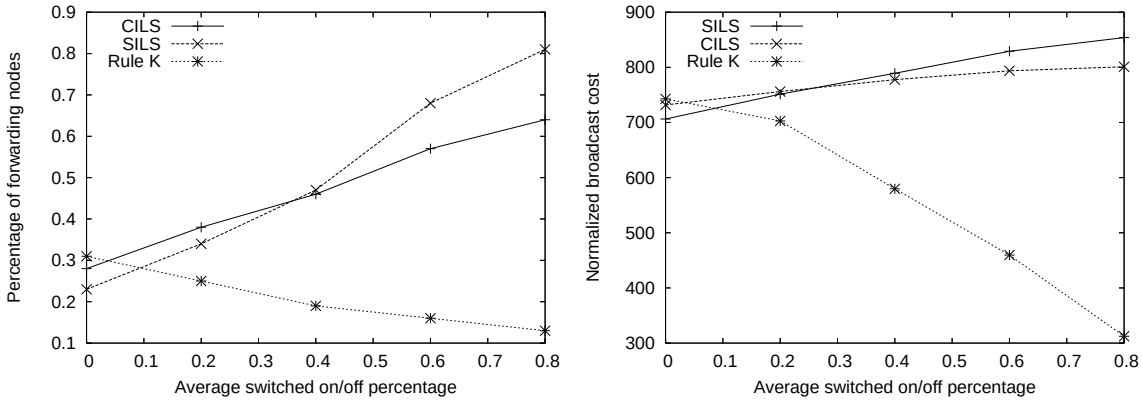
Figure 9: CDS size at each round in the switched-on/off model ($n = 100, k = 10$).

algorithm has competitive performance in unit disk graphs and we use the result of MCDS as a baseline in the comparison.

To generate a random network, n nodes are randomly placed in a restricted 1000×1000 area. The network is modeled as a unit disk graph with a fixed transmission range of 250. Networks that cannot form a connected graph are discarded. The tunable parameters in this simulation are as follows: (1) The node number n, which changes from 80 to 150, and (2) the iteration number (round number) k. The performance metric is the number of nodes in the resultant connected dominating set (CDS). For each tunable parameter, the simulation is repeated 1000 times or until the confidence interval is sufficiently small ($\pm 1\%$, for the confidence level of 90%). We use the restricted Rule K with $h = 2$ as a sample local solution for ISL.

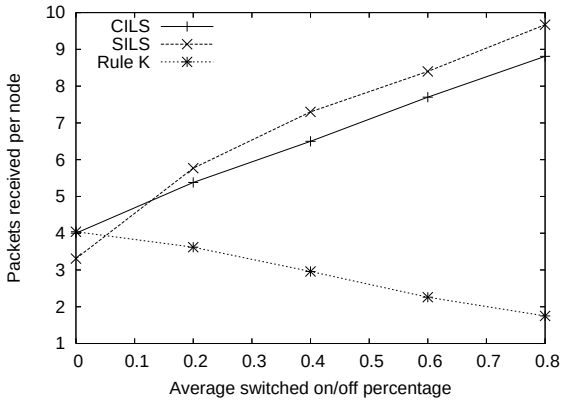
Figure 8 (a) shows the comparison of Rule $1\&2$, Rule K, LPC, and several implementations of ILS with priorities of shifting scheme (Shifting), random node value (Random), perfect shuffle (Shuffle), and MCDS, where the iteration number k equals 8. We can see that LPC beats non-iterative Rules 1&2 and Rule K, and ILS has even smaller CDS size than LPC. Among the three node priority approaches, Random has the best performance. Shuffle is better than Shifting when the number of nodes is relatively large. Since Random has the best performance, we use this approach for ILS in the subsequent analysis.

Figure 8 (b) shows the results of ILS with different iteration numbers (k). We can see that, with a larger k, the size of the resultant CDS is smaller. But when k increases to 10, the performance can

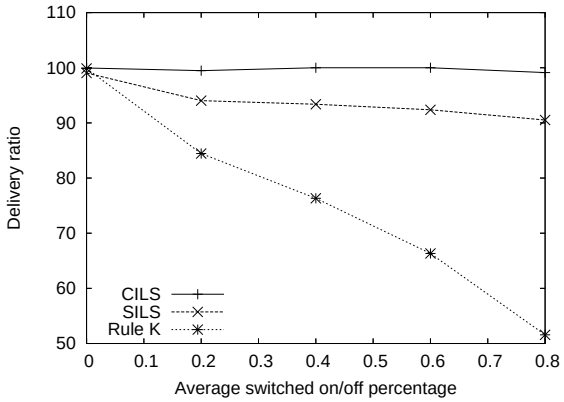


(a) Percentage of forward nodes

(b) Normalized broadcast costs



(c) Packets received per node



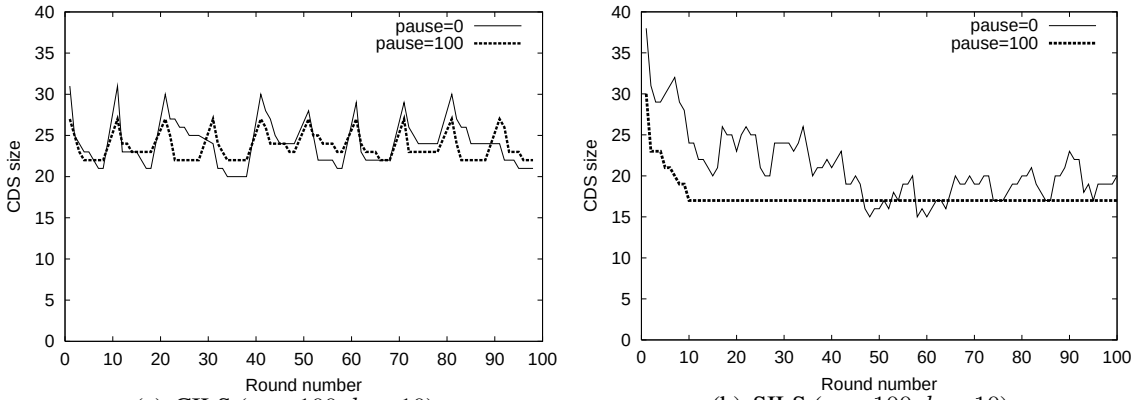
(d) Delivery ratio

Figure 10: Analysis with different switched-on/off percentage ($n = 100, k = 10$).

hardly be further improved. Therefore, we use $k = 10$ in the following simulation.

6.2 Dynamic environment

In this subsection, cyclic iterative local solution (CILS) and seamless iterative local solution (SILS) are evaluated in a WSN environment, where the topology change is caused by nodes switching on and off or movement. The simulation in this subsection is conducted on ns2 and its CMU wireless and mobility extension [20], using the IEEE 802.11 MAC layer, limited queue space in the link layer, and the two-ray ground reflection radio propagation model with constant antenna heights. An ad hoc or sensor network is also a dynamic unit disk graph under this model. The simulation time is $100s$ and



(a) CILS ($n = 100, k = 10$) (b) SILS ($n = 100, k = 10$)

Figure 11: CDS size at each round in the random waypoint model.

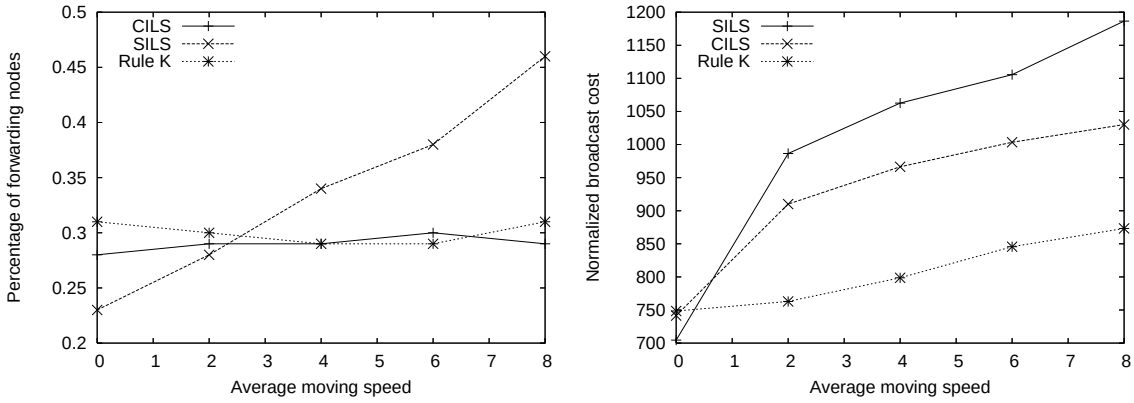
the traffic load is 1 broadcast packet per second.

The switched-on/off model. We design and implement the switched-on/off model in *ns2*. In the switched-on/off model, only a subset of deployed nodes is active. After every fixed pause time ($10s$), a certain amount of nodes (determined by the switched on/off percentage) change their status (from on to off or vice versa). The switched on/off percentage which represents the significance of the topology change, is a tunable parameter.

Figure 9 (a) demonstrates the CDS size at each round of CILS in a single run of this simulation. Two percentages of switched-on/off nodes, 0 and 0.01, are used to represent static and dynamic environments. The result is consistent with the theoretical analysis. There is little difference whether the network is static or dynamic. In each cycle, which is 10 rounds (i.e., $k = 10$), the CDS size decreases. In the next cycle, all the working nodes are marked again. Therefore, even though there is no topology change, the CDS size jumps up at the beginning of each cycle.

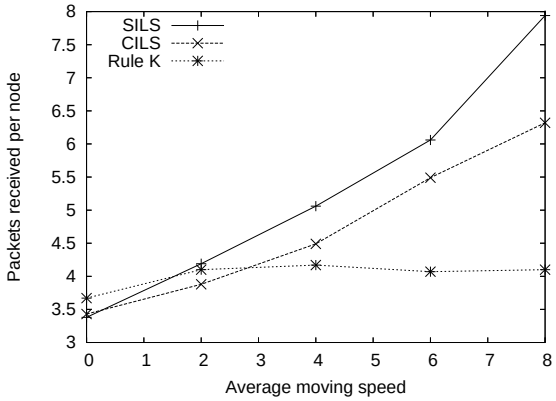
Figure 9 (b) shows the CDS size at each round of SILS in a single run. When the network is static, the CDS size decreases with rounds, achieves minimum at about round 5, and stays there. When the network is dynamic, the CDS size oscillates with rounds. But since the topology change is not significant and SILS has better locality, the oscillation is less than that of CILS.

Figure 10 shows the performance of CILS, SILS and Rule K with different switched on/off percentages. (a) shows the average percentage of forwarding nodes in the network, which represents the CDS size. We can see that when the switched on/off ratio is 0, that is, the network is static, SILS has

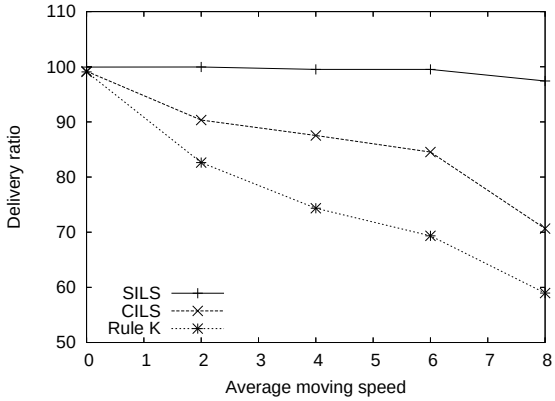


(a) Percentage of forward nodes

(b) Normalized broadcast costs



(c) Packets received per node



(d) Delivery ratio

Figure 12: Analysis with different average moving speed ($n = 100, k = 10$).

the smallest CDS size and Rule K has the largest. SILS is better than CILS because CILS calculates the CDS periodically, as shown in Figure 6, which leads to larger average CDS size. With the growth of the switch on/off ratio, both CILS and SILS have increasing CDS size while Rule K has decreasing size. This is because Rule K only calculates the CDS once. With less active nodes, its CDS size is smaller. SILS has a larger CDS than CILS when the network change is significant, because SILS maintains the locality property as topology changes and maintains a CDS at each round.

Figure 10 (b) shows the normalized broadcast costs of the three protocols with different switched on/off percentages. (c) shows the packets received per node. We can see that these two results are similar to the results in (a). This is because larger CDS size leads to larger broadcast overhead. (d) is the delivery ratio comparison. The delivery ratio of SILS decreases very slightly when the topology

changes. CILS has the second best performance and Rule K has the worst.

The random waypoint mobility model. In the random waypoint mobility model [7], each node selects its destination randomly within the deployment region and moves with a random speed. When it reaches the destination, the node pauses for a fixed time period, $10s$, and then repeats this process. The average moving speed is a tunable parameter which represents the magnitude of the topology change.

Figure 11 (a) shows the size of resultant CDS at each round of CILS, and Figure 11 (b) is that of SILS in a single run. Similar to the result shown in Figure 9, CILS has oscillation regardless of whether the nodes move or not. SILS is stabilized in round 10 when there is no movement; it oscillates when there is movement, but the oscillation is calmer and the size of CDS is smaller than that of the first several rounds.

Figure 12 shows the performance of CILS, SILS, and Rule K with different average node velocities. (a) is the percentage of forwarding nodes, which represents the CDS size. (b) is the normalized broadcast cost. (c) is the packets received per node and (d) is the delivery ratio. We can see that these figures are quite similar to those in Figure 10. When the network is relatively static, SILS generates the smallest CDS and CILS has the second best performance. Under high mobility, Rule K fails to find the valid CDS, and CILS has a smaller average CDS size than SILS but the resultant CDSs during the cycle may be broken.

The simulation results can be summarized as follows:

1. The proposed iterative local solution for CDS, using the restricted Rule K as the local solution, has better performance than Wu and Li's Rule $1\&2$, Dai and Wu's Rule K, and Liu, Pan, and Cao's algorithm in terms of the resultant CDS size.

2. The iteration number k of ILS does not need to be large to achieve good performance; $k = 10$ is sufficient in our setting.

3. Among the three node priority selection approaches, random node value has the best performance in terms of CDS size.

4. In a dynamic environment, in either the switched-on/off or random waypoint model, SILS has better performance than CILS. SILS responds quickly (locality) to network topology change while CILS does not respond to the change during a cycle.

5. When the network is relatively static, CILS has larger average DS size than SILS. Note that CILS does not guarantee a CDS while SILS does.

7 Conclusions

This paper provides a general framework for the iterative local solution. The main contribution is the seamless integration of the iterative process and the handling of topology changes in ad hoc wireless networks which include both WSNs and MANETs. We have considered two extensions to the iterative local solution to extend its use beyond the static environment. One is a natural extension that fails to obtain many desirable properties. The other uses monotonically increasing node priority to achieve seamless integration and maintain several desirable properties. The work of this paper provides insights on how to add some new features to a typical local solution in a dynamic environment.

The iterative local solution is not restricted to calculating CDS, but can apply to other applications as well, such as network topology control in MANETS and area coverage in WSNs. The extension of the current framework to cover a wide range of applications in WSNs and MANETs will be our future research direction.

References

[1] C. Adjih, P. Jacquet, and L. Viennot. Computing connected dominated sets with multipoint relays. Technical Report 4597, INRIA-Rapport de recherche, Oct. 2002.

[2] C. Adjih, P. Jacquet, and L. Viennot. Computing connected dominated sets with multipoint relays. *http://www.inria.fr/rrrt/rr-4597.html*, 2002.

[3] K. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Trans. of Parallel and Distributed Systems*, 14(5):408–421, 2003.

[4] K. M. Alzoubi, P. J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. *Proc. of 3rd ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'2002)*, pages 157–164, 2002.

[5] K. M. Alzoubi, P. J. Wan, and O. Frieder. New distributed algorithm for connected dominating set in wireless ad hoc networks. *In Proc. of 35th Hawaii Int'l Conf. on System Sciences (HICSS-35)*, 2002.

[6] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli. Localized protocols for ad hoc clustering and backbone formation: A performance comparison. *IEEE Trans. of Parallel and Distributed Systems*, 17(4):292–306, Apr. 2006.

[7] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communication & Mobile Computing (*WCMC*): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[8] J. Carle and D. Simplot-Ryl. Energy-efficient area monitoring for sensor networks. *IEEE Computer*, 37(2):40–46, 2004.

[9] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. SPAN: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proc. of 7th ACM MOBICOM*, 2001.

[10] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. SPAN: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks Journal*, 8(5):481–494, 2002.

[11] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operation Research*, 4(3):233–235, 1979.

[12] T. Clausen and P. Jacquet. Optimized link state routing protocol. IETF drafts (draft-ietf-manet-olsr-11.txt), 2003.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw Hill, 2001.

[14] F. Dai and J. Wu. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Transaction on Parallel and Distributed Systems*, 15(10):908–920, 2004.

[15] B. Das, R. Sivakumar, and V. Bharghavan. Routing in ad-hoc networks using a spine. In *Proc. of IC3N*, 1997.

[16] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. In *Proceedings of ACM-SIAM SODA*, pages 717–724, Jan. 2003.

[17] K. Fall and K. Varadham. The *ns* manual. *The* VINT *Projest, http://www.sis.edu/nanam/ns/doc/*, 2002.

[18] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. In *Proc. of Symposium on Computational Geometry*, 2001.

[19] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[20] D. B. Johnson, Y.-C. Hu J. Broch, J. Jetcheva, and D. A. Maltz. The CMU Monarch project's wireless and mobility extensions to ns. In *Proc. of 42nd International Engineering Task Force*, 1998.

[21] F. Kuhn and R. Wattenhofer. Constant-time distributed dominating set approximation. In *Proc. of 22^{nd} ACM Int. Symposium on the Principles of Distributed Computing (PODC)*, 2003.

[22] H. Lim and C. Kim. Flooding in wireless ad hoc networks. *Computer Communications Journal*, 24(3-4):353–363, 2001.

[23] C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1265–1275, 1996.

[24] H. Liu, Y. Pan, and J. Cao. An improved distributed algorithm for connected dominating sets in wireless ad hoc networks. In *Proc. of International Symposim on Parallel and Distributed Processing and Applications (ISPA), Lecture Notes in Computer Science*, number 3358, 2004.

[25] W. Lou and J. Wu. On reducing broadcast redundancy in ad hoc wireless networks. *IEEE Transactions on Mobile Computing*. 1, (2), April-June 2002, 111-122.

[26] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25:59–68, 1995.

[27] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast message in mobile wireless networks. In *Proc. of 35th Hawaii Int'l Conf. on System Sciences (HICSS-35)*, 2002.

[28] P. Sinha, R. Sivakumar, and V. Bharghavan. CEDAR: a core-extraction distributed ad hoc routing algorithm. *IEEE Journal on Selected Areas in Communications, Special Issue on Ad Hoc Networks*, 17(8):1454–1465, Aug. 1999.

[29] P. Sinha, R. Sivakumar, and V. Bharghavan. Enhancing ad hoc routing with dynamic virtual infrastructures. In *Proc. of IEEE INFOCOM'2001*, 2001.

[30] I. Stojmenovic. Data gathering and activity scheduling in ad hoc and sensor networks. In *Proc. of International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peer-to-Peer Networks*, 2004.

[31] I. Stojmenovic, S. Seddigh, and J. Zunic. Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks. *IEEE Trans. on Parallel and Distributed Systems*, 13(1):14–25, 2002.

[32] H. S. Stone. *High-performance computer architecture*. Addison-Wesley, 3rd edition, 1993.

[33] P. J. Wan, K. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *Proc. of IEEE INFOCOM'2002*, 2002.

[34] J. Wu and F. Dai. A generic distributed broadcast scheme in ad hoc wireless networks. In *Proc. of ICDCS'2003*, 2003.

[35] J. Wu and H. Li. On calculating connected dominating sets for efficient routing in ad hoc wireless networks. In *Proc. of ACM DIALM'99*, 1999.

[36] J. Wu and W. Lou. Forward-node-set-based broadcast in clustered mobile ad hoc networks. *Wireless Networks and Mobile Computing, a Special Issue on Algorithmic, Geometric, Graph, Combinatorial, and Vector Aspects*, 3(2):155–173, 2003.