Discrete Mathematics, Algorithms and Applications
Vol. 1, No. 4 (2009) 437–461
© World Scientific Publishing Company

EFFICIENT DISTRIBUTED ALGORITHMS FOR TOPOLOGY CONTROL PROBLEM WITH SHORTEST PATH CONSTRAINTS*

JAMES K. WILLSON[†,§], XIAOFENG GAO[†,¶], ZHONGHUA QU[†,‖], YI ZHU[†,**], YINGSHU LI[‡,††] and WEILI WU[†,‡‡]

[†]*Department of Computer Science, University of Texas at Dallas*
2601 North Floyd Road, Richardson, TX 75083, USA
[‡]*Department of Computer Science, Georgia State University, USA*
[§]*jkw053000@utdallas.edu*
[¶]*xiaofenggao@utdallas.edu*
[‖]*zxq071000@utdallas.edu*
[**]*yxz053100@utdallas.edu*
[††]*yli@cs.gsu.edu*
[‡‡]*weiliwu@utdallas.edu*

Accepted 30 November 2009

A *Connected Dominating Set* (CDS) can be used to construct a virtual backbone for wireless and mobile ad-hoc networks to make the system hierarchical and efficient. A virtual backbone can significantly improve network throughput, optimize broadband utilization, extend network lifetime, and reduce interference as well as packet retransmissions. Calculating a minimum backbone for a network is critical to reduce routing computation and energy consumption. This problem is a well-known NP-hard optimization problem, which has various applications in practice. In this paper, we propose a new problem based on customer fairness, which looks for a minimum CDS in a given communication model with shortest path constraints. It guarantees that any two clients can communicate with each other through this CDS with hop counts the same as the best path from the original graph, which means that routing on such a CDS will not bring additional traffic for every client. We name this problem as *shortest path connected dominating set* (SPCDS) and prove its NP-hardness by reduction from *Hitting Set*. Then we propose a centralized greedy algorithm and an efficient distributed approximation algorithm with approximation ratio Δ to solve SPCDS, where Δ is the maximum vertex degree in the given topology. We also analyze the time complexity, message complexity, and evaluate the efficiency of our distributed heuristic by several numerical experiments and comparisons with previous literatures.

Keywords: Connected dominating set; shortest path; distributed algorithm.

Mathematics Subject Classification 2000: 68R10, 41A99, 05C69

*Supported by the National Science Foundation under grant CCF-0514796 & CCF-0829993.
¶Corresponding author.

1. Introduction

Wireless communication networks have become increasingly popular for various applications in recent decades. In a typical wireless network, each node transmits packets through radio signals beamed within a given radio frequency radius [25]. If a receiver is located outside the transmission range of the sender, the sender can send a packet via a multi-hop path, which travels through several intermediate nodes. Generally, when a packet has been sent from one side of the link, an ACK message is required to report a successful transmission [16], so it is more convenient to operate the network if each link is bidirectional.

Since wireless networks lack physical infrastructure or dedicated devices that serve as routers, any node can serve as an intermediate router. Thus within this kind of network, finding an efficient way of routing is important, which can accomplish through *topology control.* Topology control is a strategy for energy-efficiency to reduce the cost of routing (e.g. reduce the routing table size). It can significantly improve network performance, optimize broadband utilization, extend network lifetime, and reduce interference as well as packet retransmissions [18].

There are two different types of topology control strategies. One is to select a subset of nodes as a virtual backbone in a given graph. Nodes in this backbone (dominators) are responsible for all routing work. Routing information is only kept and maintained within dominators, and other nodes (dominatees) will send packets to them. If some nodes move or leave the network, as long as they are not backbone nodes, routing is minimally disrupted. Besides, as routing information is only exchanged between backbone nodes, communication cost will be reduced heavily.

Another topology control strategy is to reduce the interference of networks, either by reducing the transmission range of each node, changing the topology of subgraphs, or assigning a schedule [26] to avoid potential packet retransmissions [5]. In this paper, we focus on the first method: finding an efficient virtual backbone in a given graph with some special constraints.

Usually, for a given communication graph G, we select a *connected dominating set* (CDS) [20] to construct a virtual backbone. The formal definition is:

Definition 1.1 (Connected Dominating Set). Given an undirected graph $G = (V, E)$, a *connected dominating set* (CDS) is a vertex subset $C \subseteq V$ satisfying:

(1) $\forall\, v \in V$, either $v \in C$, or there exists a node $u \in C$, such that $(u, v) \in E$.
(2) The subgraph induced by C (represented by $G[C]$) is connected. $G[C]$ is a subgraph with all nodes in C, and all edges in E with both ends belonging to C.

CDS is a graph model to represent a virtual backbone. Selecting a CDS with small size can make the network system hierarchical and efficient. This problem is critical to reduce routing computation and energy consumption. It is a well-known NP-hard optimization problem, which has various applications in practice.

1.1. *Previous literature*

Many researchers worked on how to find a minimum CDS (MCDS) under different situations. Garey and Johnson [12] proved that finding an MCDS is NP-hard in general graphs by a reduction from the *Set Cover* Problem. Lund [22] and Feige [9] showed that for any fixed number $0 < \varepsilon < 1$, there is no polynomial time algorithm with performance ratio $\leq (1-\varepsilon)H(\Delta)$ unless $NP \subset DTIME[n^{O(\log\log n)}]$, where Δ is the maximum degree of the input graph and H is the harmonic function. Baker [4] proposed a polynomial time approximation scheme (PTAS) for planar graphs.

Due to the homogenous nature of wireless networks, numerous previous works consider their communication model under *unit disk graph* (UDG) [11, 29], a kind of graph where each node has the same broadcasting radius and two nodes can communicate if they are located within a fixed communication range of each other. Clark [8] proved that *minimum dominating set* (MDS) remains NP-hard in UDG. Lichtenstein [21] showed that MCDS is NP-hard in UDG. Many prior works provided constant-factor approximation for MCDS in UDG [1, 2, 6, 11, 13, 14, 23, 24]. Cheng *et al.* [7] and Hunt *et al.* [15] gave PTAS for computing a MCDS in UDG.

Later people generalized this model into *disk graph* (DG) [20, 27, 28], in which nodes can have different transmission ranges, and each edge may not be bidirectional. Zhang *et al.* [32] further extend this model into *unit ball graph* (UBG), a generalized UDG in 3-dimensional space. For other variations, Ambuhl [3], Gao [10] proposed a constant-factor approximation for the weighted CDS problem. Say, in their graph model, each node has different weight and the goal is to find a minimum weight CDS (MWCDS). Some researchers also think about deploying some new nodes to dominate the original network [17].

1.2. *Problem statement*

Although CDS is an efficient virtual backbone for routing protocols, if every node sends packets through this backbone, the routing path between some pairs of nodes might increase greatly. Figure 1 is an example showing this case. In this figure, given a graph G, if v_1 wants to send a packet to v_7, it only needs to send the packet through a path with 2-hop length, $v_1 \rightarrow v_6 \rightarrow v_7$, shown as black arrows in Fig. 1(a). However, if we select a CDS for G (shown as grey nodes in Fig. 1(b)), v_1 must send

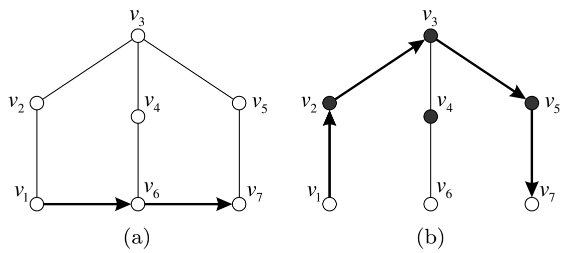


Fig. 1. Unfairness for clients: (a) traffic in original topology; (b) traffic through CDS.

packets via this CDS, which is a 4-hop path to v_7 ($v_1 \to v_2 \to v_3 \to v_5 \to v_7$). The second path is twice as long as the first one. This example shows that CDS may cause additional traffic for some clients, which creates unfairness and inefficiency. However, if we insert v_6 into this CDS, the total size of CDS only increases by one, but the total traffic for v_1 will reduce significantly. Therefore, we have a trade-off between CDS size and client fairness.

Previous literature provided some results to deal with this constraint. Wu [31] introduced a simple distributed algorithm to construct a CDS with *all shortest paths property*, that is, all the intermediate nodes along all shortest paths from u to v are included in this CDS. Each node can decide if it is part of the CDS given information of its 2-hop neighborhood. This takes $O(\Delta^2)$ time and $O(\Delta v)$ messages, where Δ is the maximum node degree in this communication graph.

They also provided pruning policies to deal with redundant nodes. Node u can be removed from the CDS if it has a neighbor with greater *id* which covers all the other nodes u covers, or if it has two neighbors with greater *id* which together cover all nodes u covers. The pruned CDS may be smaller, but the all shortest paths property is not maintained. And even with this extension, Wan [29] proves their approximation ratio is $O(n)$.

Wang [30] described a distributed algorithm for constructing a k-spanner. They constructed a subgraph such that the shortest path between any two nodes is at most a constant factor larger than the shortest path in the original graph. The algorithm attempted to construct a spanner with few links and low node degree but the number of nodes that must participate in routing may be large.

1.3. *Our contribution*

The goal for this paper is to find a CDS with small size, such that for any pair of nodes, this CDS offers at least one path which has the same length as the shortest path from the original graph. We name this problem as *Shortest Path Connected Dominating Set* (SPCDS). We prove its NP-hardness by reduction from *Hitting Set*, then prove that SPCDS is equivalent to another optimization problem named SPP-2. Next, we proposed an efficient centralized greedy heuristic for SPP-2. Since in wireless ad-hoc networks, each node will make its decisions independently according to their local information, we also design a distributed approximation for SPP-2. We provide correctness, time complexity, message complexity, and approximation ratio for these two algorithms and then evaluate their efficiency by several simulations. We compare the output with previous works in [31], which demonstrates that the size of our solution is smaller. Our algorithm is the first distributed approach to select a CDS which guarantees fairness for clients.

The rest of this paper is organized as follows. Section 2 provides some preliminaries and shows the equivalence of SPCDS and SPP-2. In Sec. 3, we propose a centralized algorithm to solve SPP-2, with performance analysis and detailed discussion. Section 4 describes a distributed version of the algorithm for SPP-2, with the help of a coloring procedure. In this section we also analyze the correctness, time complexity, message complexity and approximation ratio for this algorithm. Section 5 provides several numerical experiments to evaluate our algorithm. We test our algorithm in different environments, and provide a comparison with optimal solutions in small scale networks, and another comparison with Wu's work [31] in large scale networks. Finally, Sec. 6 gives the conclusion and future work.

2. Preliminaries

In this section, first we list several assumptions needed to model our problem, and then predigest this problem into an equivalent problem, with several related assumptions, concepts, and theorems.

2.1. *Assumptions*

We have the following assumptions for our communication model:

- The topology is stable during a period of time.
- We consider an arbitrary undirected graph.
- The distributed system is asynchronous.
- The network is homogenous. (Or the graph does not have weight function.)
- There is no failure in the system (e.g. no crashes).

2.2. *Problem predigestion*

We firstly define our problem named *shortest path connected dominating set* (SPCDS), which includes the following requirements.

Definition 2.1 (SPCDS). Given an undirected connected graph $G(V, E)$ where V and E are the node and edge set of the graph, respectively. The *shortest path connected dominating set* (SPCDS) problem is to find a set $S \subseteq V$ such that

(1) S is a dominating set;
(2) the induced graph $G[S]$ is connected;
(3) S satisfies the shortest path property — for any two nodes u, v, there is a shortest path from u to v all of whose intermediate nodes are in S.

Note that for one-hop paths, we will not consider the shortest path property (e.g. for a path (u, v), S may not include both u and v), since nodes at each side can communicate directly.

We can see that SPCDS has three requirements. However, the first and second requirements are superfluous. From the following analysis, we prove that they can be removed so long as G is not a complete graph.

Lemma 2.2. *If the shortest path property holds for $S \subseteq V$, and G is not a complete graph, then $S \neq \emptyset$.*

Proof. If G is not a complete graph, then $\exists\, u, v \in G$ such that u and v are not neighbors. By the shortest path property, S must contain the intermediate nodes of a shortest path from u to v. Thus $S \neq \emptyset$. $\square$

Lemma 2.3. *If the shortest path property holds for $S \subseteq V$, and $S \neq \emptyset$ then S is a dominating set.*

Proof. We pick up any $u \in V$, and $u \notin S$. $S \neq \emptyset$, so there exists some $v \in S$. If u and v are neighbors, then u is dominated. If u and v are not neighbors, since G is connected, there is some shortest path from u to v in G. By the shortest path property, the intermediate nodes of at least one such shortest path are in G, denoted as $P(u, v) = \{u, p_1, p_2, \ldots, p_t, v\}$. $p_i \in S$, $\forall 1 \leq i \leq t$. Then the first intermediate node p_1 dominates u. Since any arbitrary node not in S is dominated, S is a dominating set. $\square$

By Lemmas 2.2 and 2.3, S is dominating if the shortest path property holds. Thus we can delete condition (1) in Definition 2.1.

Lemma 2.4. *If the shortest path property holds for set S, $G[S]$ is a connected graph.*

Proof. We must show $\forall\ u, v \in S$, there is a path from u to v using only nodes in S. Such a path is guaranteed by shortest path property. Thus $G[S]$ is connected. $\square$

By Lemma 2.4, we can delete (2) in Definition 2.1. Then we get Theorem 2.5.

Theorem 2.5. *If G is not a complete graph and S satisfies the shortest path property* (*condition* 3), *S is a solution for* SPCDS.

Since we are aiming at distributed algorithms, next we prove that SPCDS problem is equivalent to another problem, named SPP-2.

Definition 2.6 (SPP-*k*). Given a graph $G = (V, E)$, let $P_S(u, v)$ denote the shortest path between nodes u and v using only intermediate nodes in S, where $S \subseteq V$. $|P_S(u, v)|$ is the length of this shortest path. A node subset $S \subseteq V$ has the shortest path property for length k (SPP-k) if $\forall\, u, v \in V$ with $|P_V(u, v)| \leq k$, we have $|P_S(u, v)| = |P_V(u, v)|$. The SPP-$k$ problem is to find a minimum $S \subseteq V$ such that S satisfies the SPP-k property.

Here SPP-k property is the shortest path property for paths of length k or less. It is easy to see that SPP-2 is a special case of SPP-k where $k = 2$. In SPP-2, we are aiming at selecting a smallest subset $S \subseteq V$ such that $|P_S(u, v)| = |P_V(u, v)|$ for every u, v with $|P_V(u, v)| \leq 2$, which means that u and v are connected by only one intermediate node.

Theorem 2.7. SPCDS *is equivalent to* SPP-2.

Proof. We use induction to prove the equivalence. Let D denote the diameter of graph G. It is trivial if S satisfies SPP-D, S is a solution of SPCDS. Next, let's prove that if S satisfies SPP-2, then S satisfies SPP-k for any integer $k > 2$.

Assume for induction that SPP-n holds for S, $n \geq 2$. Pick up any node pair $u, v \in V$ with $|P_V(u,v)| = n+1$, then there is a path of length $n+1$ from u to v written as $P_V(u,v) = \{u, p_1, p_2, \ldots, p_n, v\}$. By the inductive hypothesis, there is a path of length n from u to p_n, with intermediate nodes $x_1, x_2, \ldots, x_{n-1}$; $x_i \in S$, $\forall 1 \leq i \leq n-1$. Since x_{n-1} and v are both neighbors to p_n, there is a path of length 2 between them. Furthermore, x_{n-1} and v cannot be neighbors, otherwise there is a path of length n from u to v. By the inductive hypothesis, there is a path of length 2 between x_{n-1} and v with intermediate node $x_n \in S$. Therefore, $x_1, x_2, \ldots, x_n$ are intermediate nodes in S along a path of length $n+1$ from u to v. So SPP-$(n+1)$ holds. This construction is shown in Fig. 2. By induction, if SPP-2 holds, then SPP-k holds. Thus SPCDS problem is equivalent to SPP-2 problem. □

Theorem 2.8. SPCDS *problem is* NP-*hard.*[a]

In the following sections, we will focus on algorithms for SPP-2, since its solution will fit the requirements of SPCDS.

3. A Greedy Algorithm for SPP-2

3.1. *Algorithm description*

In this section, we will introduce a simple centralized greedy algorithm to solve SPP-2. The idea is that we first consider all nodes in V as the required nodes, then iteratively remove redundant nodes. The algorithm can be described in Algorithm 1.

To evaluate whether S–$\{v\}$ satisfies SPP-2 in Line 6, we only need to check the neighbors of v, to see whether they can find a path through S–$\{v\}$ with 2 hops.

3.2. *Performance analysis*

In this subsection, we will go over GSPP-2 in detail by analyzing the performance including correctness, time complexity, and approximation ratio.

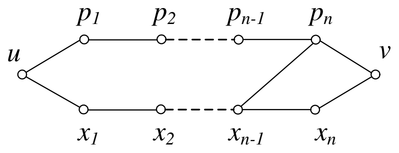


Fig. 2. Proof for Theorem 2.7.

[a]We prove this theorem by a reduction from *Hitting Set*, which can be seen in Appendix.

Algorithm 1 Selecting Node Set Satisfies SPP-2 (GSPP-2)

```
1: Input: an undirected connected network $G(V, E)$, where $V$ and $E$ are the node
   and edge set of the graph $G$, respectively.
2: Output: a node set $S \subseteq V$ which satisfies SPP-2.
3: Begin
4:     $S \leftarrow V$
5:     for all $v \in V$
6:         if $S - \{v\}$ satisfies SPP-2
7:         $S \leftarrow S - \{v\}$
8:         endif
9:     endfor Output $S$.
10: End
```

Theorem 3.1. GSPP-2 *selects a minimal node set satisfying* SPP-2.

Proof. First let us prove that the output S is feasible. Initially, $S = V$, so it must be feasible. It is a loop invariant that S remains feasible. Thus the output S is feasible. Next, we show that S is minimal. If not, then some $v \in S$ can be removed, but then v would have been removed when it was encountered in the loop. Thus, Theorem 3.1 is true based on the selection procedure of GSPP-2. □

Theorem 3.2. GSPP-2 *has time complexity* $O(|V| \cdot \Delta^3)$, *where* Δ *is the maximum degree in* G.

Proof. For every node v in V, we need to check its 1-hop neighborhood $N(v)$, to see whether S–$\{v\}$ provides equivalent shortest path for v's neighbors. This step requires at most $O(\Delta^3)$ time (for each pair of v's neighbors, it needs $O(\Delta)$ time to check a path, and there are at most Δ^2 pairs), and totally we need to check $|V|$ nodes. Therefore the time complexity of GSPP-2 is $O(|V| \cdot \Delta^3)$. □

Theorem 3.3. GSPP-2 *has approximation ratio* Δ.

Proof. Let $G = (V, E)$ be an instance of SPCDS, S_{opt} be an arbitrary optimum solution, S_{alg} be an arbitrary minimal solution selected from our algorithm. We partition V into four sets as follows:

B: $S_{\text{opt}} \cap S_{\text{alg}}$ (nodes in both solutions)
P: $S_{\text{opt}} \backslash S_{\text{alg}}$ (nodes only in the optimum solution)
M: $S_{\text{alg}} \backslash S_{\text{opt}}$ (nodes only in the minimal solution)
N: all remaining elements of V (nodes in neither solution)

If $P = \emptyset$, then by the minimality of S_{alg}, $S_{\mathrm{alg}} = S_{\mathrm{opt}}$. Otherwise, let M_i be those nodes $u \in M$ such that the closest node $v \in P$ is i hops away. Then we will have:

(1) $M_k = \emptyset$ for all $k \geq 3$.

Suppose for contradiction such a node u exists. By minimality of S_{alg}, there is some pair of nodes a and b such that u is their only common neighbor in S_{alg}, and hence $B \cup M$. Since $u \in M$, a and b share no common neighbor in B. Since u is at least three hops away from any node in P, a and b share no common neighbor in P. So a and b have no common neighbor in S_{opt}, contradicting the feasibility of S_{opt}.

(2) If $u \in M_2$ then u has a neighbor $v \in B$, and v has a neighbor $w \in P$.

Pick up any $u \in M_2$. Then there is a path of length 2 from u to some node $w \in P$, and u and w are not neighbors. By the feasibility of S_{opt}, u and w must share a common neighbor in B or P. Since $u \in M_2$, u has no neighbor in P. So u and w share a neighbor $v \in B$.

Since all nodes in M_2 have a neighbor in B, and itself has a neighbor in P, $|M_2| \leq (\Delta - 1)|B|$; all nodes in M_1 have a neighbor in P, $|M_1| \leq \Delta|P|$. Thus,

$$\begin{aligned} |S_{\mathrm{alg}}| &= |B| + |M| = |B| + |M_1| + |M_2| \leq |B| + \Delta|P| + (\Delta - 1)|B| \\ &= \Delta(|P| + |B|) = \Delta|S_{\mathrm{opt}}|, \end{aligned}$$

which implies that GSPP-2 is a Δ-approximation. □

4. A Distributed Algorithm for SPP-2

In this section, we discuss how to implement GSPP-2 in a distributed manner. Here two observations provide the basis of the distributed version, Algorithm 2.

The first observation is that when deciding whether S–$\{v\}$ satisfies SPP-2, we do not need to know S in its entirety. Instead, v's 2-hop neighborhood is sufficient. Removing v from S can only cause SPP-2 to fail if there is some path of length 2 with v as an intermediate node, and no other node in S fills that role. Any such path will appear in v's 1-hop neighborhood, and all alternate paths will show up in v's 2-hop neighborhood.

The second observation is that the nodes need not be tested strictly sequentially. For instance, if two nodes u and v share two common neighbors a and b, where a and b are not neighbors, then we say that u and v interfere with each other. One of the common neighbors of a and b must be in S to maintain SPP-2. However, if u and v do not interfere, their status in S can be decided simultaneously. That is, if S-$\{u\}$ is safe, and S–$\{v\}$ is safe, and u, v do not interfere, then S-$\{u, v\}$ is safe. Any two interfering nodes will be separated by at most two hops. We say $(V, \{(u, v)|u \text{ and } v \text{ interfere}\})$ is the *interference graph.* Figure 3 is a simple example of an interference graph. Figure 3(a) is the original topology for four nodes, and Fig. 3(b) is the interference graph. (u, v) exists in the interference graph because they neighbor both a and b, and a and b are not neighbors.

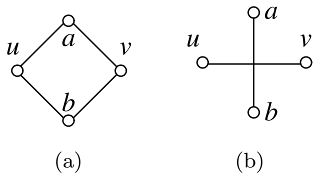


Fig. 3. An example interference graph.

4.1. *Algorithm description*

Based on two observations, the main idea of Algorithm 2 can be described as follows.

First, each node constructs its 2-hop neighborhood from the 1-hop neighborhoods of its neighbors. From this, it determines which nodes it interferes with. Second, every node v should decide whether $S-\{v\}$ is feasible. The nodes do not have to do this step in a strict sequential order, but neither can they all execute it at once. So we are going to use coloring to schedule the execution of nodes. Any distributed graph coloring algorithm can be used to color the nodes of the interference graph. (Neighbors in the interference graph might not be neighbors in G, so an intermediate node may need to relay messages between interfering nodes.)

A simple distributed graph coloring algorithm is described in [19]. In each round, every uncolored node chooses a number from 0 to Δ, the degree of the node. The node may not choose the color of any of its colored neighbors. If a node chooses a color not chosen by any of its neighbors that round, the node keeps it. Otherwise, the node remains uncolored.

Once a node and all its neighbors in the interference graph are assigned colors, the node waits for higher priority interference neighbors (as determined by the

Algorithm 2 Distributed Selection for SPP-2 (DSPP-2)

```
1:  Input: an undirected connected network $G(V, E)$, where $V$ and $E$ are the node
    and edge set of the graph $G$, respectively.
2:  Output: A node set $S \subseteq V$ satisfying SPP-2.
3:  Begin
4:      Send 1-hop neighborhood information to all neighbors.
5:      Wait to receive 1-hop neighborhood information from all neighbors.
6:      Calculate neighbors in the interference graph.
7:      Color the interference graph.
8:      Wait for decisions from all interference neighbors with higher priority.
9:      If $v$ not being in the set is consistent with higher priority neighbor decisions
    and SPP-2, then $v$ decides itself not to be part of the set; else $v$ decides itself
    to be part of the set.
10:     Send decision to all interference neighbors with lower priority.
11: End
```

coloring). Once those nodes have decided whether to be in or out of S, the node makes its own decision. Say, it checks if it has a pair of neighbors a and b, a and b not themselves neighbors, and all other common neighbors of a and b are nodes which have decided not to be in S. If so, the node decides to be in S. Otherwise, it decides not to be in S. It then informs its lower priority interference neighbors of its decision. Algorithm 2 shows the details. Each node v will run a copy.

The result set includes all nodes which decide themselves to be part of the set.

4.2. *One scenario for DSPP-2*

In this subsection, we show one possible running scenario for DSPP-2. Figure 4 is an example for such a run. The left figure is the original topology of graph G, including 12 nodes and 17 edges. The right figure is the induced interference graph for G, we name it as G' (This is calculated from Line 4 to Line 6 in DSPP-2).

Next, we will run the coloring algorithm and get one potential result for G'. Figure 5 illustrates the process of the coloring step. The left figure is the potential color numbers for each node, which is correlated to the node degree. For instance, [0-2] in Fig. 5(a) means that this node can choose its color from $\{0, 1, 2\}$. Next, after running the coloring algorithm, we can have one possible coloring result as shown in Fig. 5(b). Here we choose color number from 0 to 3, and these numbers denote different colors: 0 is blue, 1 is yellow, 2 is red and 3 is black (This step is described in Line 7 in DSPP-2). Let 0 be the highest priority, and 3 the lowest.

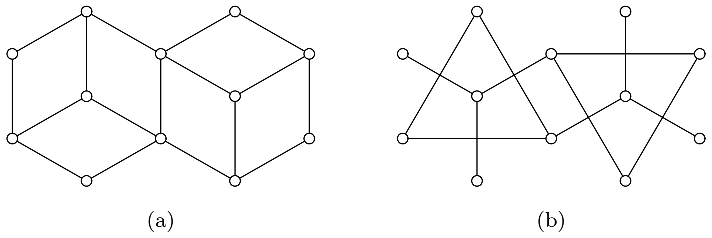


Fig. 4. Scenario for DSPP-2: (a) Topology; (b) Interference graph.

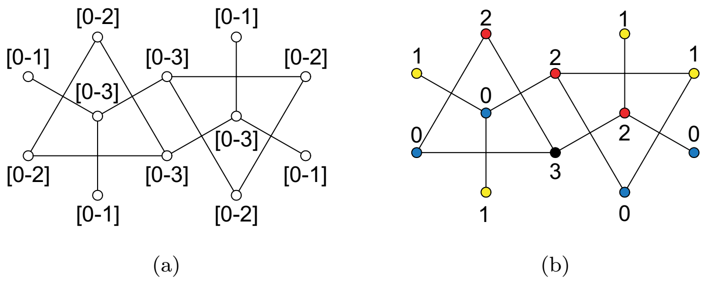
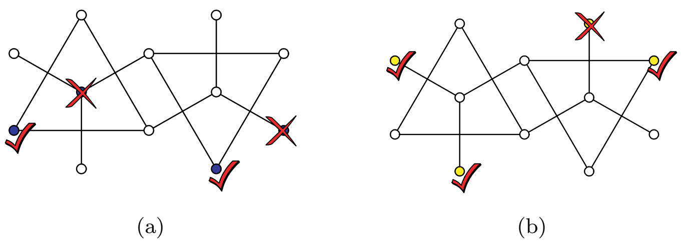


Fig. 5. (Color online) (a) Color selection range; (b) Potential color selection.



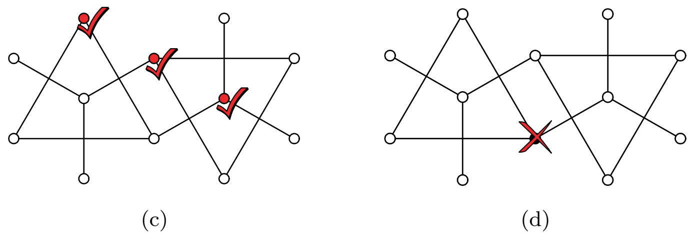


Fig. 6. (Color online) (a) Decision for blue nodes; (b) decision for yellow nodes; (c) decision for red nodes; (d) decision for black nodes.

After the coloring process, we have the priority order for each node. Then they can decide whether they belong to target set S independently according to priorities. Figure 6 shows the results for each priority. Figure 6(a) illustrates decision process for nodes with priority 0 (blue color); after all blue nodes make their decision, nodes with priority 1 (yellow color) will make decisions (shown in Fig. 6(b)); next, nodes with priority 2 and 3 will run this algorithm in turn (shown in Figs. 6(c) and 6(d)). Finally, all "checked" nodes will be selected in S as our final result. This process is described from Line 8 to 10. Note that DSPP-2 is asynchronous, so node v can start its run when all its neighbors with higher priority have made their decision, regardless of whether nodes with higher priority elsewhere have finished.

4.3. *Performance analysis*

In this subsection, we discuss the performance of DSPP-2, including correctness, time complexity, message complexity and approximation ratio. Let $n = |V|$, $m = |E|$, $\Delta =$ maximum degree of G. In the following, we assume message sizes of at most $O(\lg n)$ to make the time complexity and message complexity discussion meaningful (it means that node identifier takes space $O(\lg n)$).

Theorem 4.1. DSPP-2 *has expected time complexity* $O(\Delta \lg n + \Delta^3)$ *and message complexity* $O(m\Delta \lg n)$.

Proof. We can divide DSPP-2 into three parts and analyze the time complexity and message complexity separately for each part.

Part 1: Part 1 is Lines 4 to 6 in DSPP-2, which describe how to gather 2-hop neighborhood information and calculate the interference graph. It takes $O(\Delta)$ time and $O(m\Delta)$ messages. For any two neighboring nodes u and v, u will send to v one message for each of its other neighbors (at most $\Delta - 1$).

Part 2: In Line 7 DSPP-2 colors the interference graph. In this step, we run the coloring algorithm in [19] on the interference graph induced from Part 1 (not G), so we must simulate it.

Simulating a round of the coloring algorithm takes two steps. In the first step, messages between immediate neighbors in G are sent, as well as the first hop of messages between 2-hop neighbors in G. In the second step, the second hop is sent. Given u and v, u will send at most Δ messages to v in step 1. One intended for v itself, and one to be relayed to each of v's other neighbors. In step 2, u will send v at most $\Delta - 1$ messages, relaying one message from each of u's other neighbors. So we increase the time complexity of the coloring algorithm at most by a factor of Δ.

The algorithm in [19] terminates in $O(\lg n)$ time with high probability. It assigns at most $\Delta' + 1$ colors, where Δ' is the maximum degree of the interference graph. So our simulation terminates in $O(\Delta \lg n)$ time with high probability. Trivially, it uses $O(m\Delta \lg n)$ messages with high probability.

Part 3: Lines 8 to 10 in DSPP-2 state Part 3. Since interfering nodes are at most two hops apart, the maximum degree of the interference graph (Δ') is no more than the largest 2-hop neighborhood. That, in turn, is bounded by Δ^2. So at most $\Delta^2 + 1$ colors are assigned to the nodes. Once all nodes are colored, nodes with color 0 will make the decision to be in or out of S. Similarly to the relaying strategy when simulating the coloring algorithm, these decisions will be reported to interference neighbors within $O(\Delta)$ time. Successive colors will be resolved with the same time bounds. There are at most $\Delta^2 + 1$ colors, so this stage of the algorithm takes $O(\Delta^3)$ time. There are $O(m\Delta)$ edges in the interference graph. We send one decision message along each edge of the interference graph, and each such message is simulated with at most two messages. So this stage uses $O(m\Delta)$ messages.

Therefore, DSPP-2 has expected time complexity $O(\Delta \lg n + \Delta^3)$ and expected message complexity $O(m\Delta \lg n)$. $\square$

Theorem 4.2. DSPP-2 *selects a minimal node set satisfying* SPP-2.

Proof. To prove the correctness of DSPP-2, we want to show that the output of DSPP-2 is a potential output of the centralized greedy algorithm GSPP-2.

Suppose we run DSPP-2. In Part 2 we choose a coloring. Then consider a run of GSPP-2, in which nodes are ordered consistent with the coloring we chose in Part 2. We order the nodes as $\{v_1, v_2, \ldots, v_t\}$. We want to argue that the outputs of these two algorithms are the same. Now we use induction to prove v_i the statement.

Assume for induction, the nodes in queue before v_i all make the same decision in GSPP-2 and DSPP-2. Now we prove that v_i makes the same decision in GSPP-2 and

DSPP-2. Consider the interference neighbors of v_i. From coloring procedure, each interference neighbor has either higher or lower priority than v_i. By the ordering of nodes, neighbors with higher priority have smaller subscript than v_i. Therefore, by inductive hypothesis, these neighbors make the same decision.

We can separate these neighbors into two sets, one includes all neighbors which decide themselves in the SPP-2 set (named as N_{in}), while the other includes neighbors which decide themselves not in the SPP-2 set (named as N_{out}). Similarly, neighbors with lower priority have greater subscript than v_i. We consider them as the third set (N_{un}). Let S_G be the value of S in GSPP-2 before we consider v_i. $N_{\text{in}} \subseteq S_G$ by inductive hypothesis. Similarly, $N_{\text{out}} \cap S_G = \emptyset$. Now we prove that $N_{\text{un}} \subseteq S_G$. That is true because neighbors with lower priority have not decided yet, so they must be in S_G. We can conclude that v_i will make the same decision in GSPP-2 and DSPP-2. □

According to Theorem 4.2, we can see that DSPP-2 has approximation ratio Δ.

5. Performance Evaluation

To measure the effectiveness of our distributed algorithm for SPP-2 (DSPP-2), we simulate DSPP-2 on several randomly generated connected topologies. Simulation results show that DSPP-2 not only produces good results, which are very close to the optimal solution in the small scale networks, but also achieves better performance than Wu's algorithms [31] (even his improved algorithm) in large scale networks.

5.1. *Simulation environment*

As mentioned above, we simulated DSPP-2 on randomly generated connected topologies. To evaluate DSPP-2 with different node densities (which represents the size of our network), we simulate this algorithm on topologies with different numbers of nodes in a square region of $100 \times 100\,\text{m}^2$. We first deploy nodes in the area randomly with uniform distribution. Then if the distance between two nodes is less than the maximum transmission range, we generate the edge between them. We need to check the connectivity after deployment. Since we consider connected dominating set, we should guarantee that the topology we generate is connected. If the result is not connected, we drop this topology and redo the deployment until we get a feasible topology.

Figure 7 shows some typical topologies for our simulation. It exhibits four topologies with different N and R values in this square region. Here N is the number of nodes in one topology, and R is the maximum transmission range. Figures 7(a) and 7(b) are examples of small scale networks, while Figs. 7(c) and 7(d) are examples of large scale networks. It is easy to see that when R and N increase, it is much easier to form a connected topology for our simulation.

To show the effectiveness of DSPP-2, we compare our results with Wu's algorithm in [31] and its improved version. For simplicity, we name the original Wu's

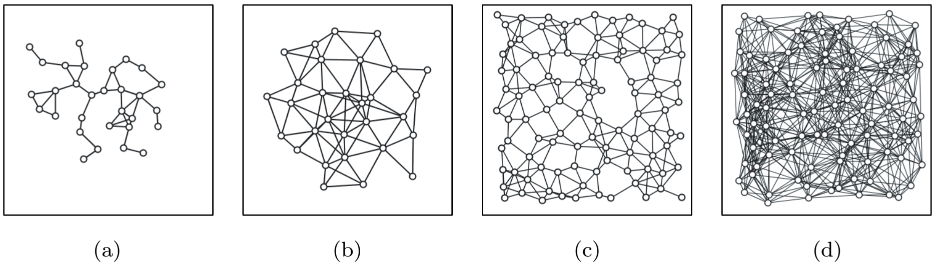


Fig. 7. Topology examples for simulations: (a) $N = 30$, $R = 10$; (b) $N = 30$, $R = 20$; (c) $N = 100$, $R = 15$; (d) $N = 100$, $R = 30$.

algorithm as Wu-Sim algorithm, and Wu's algorithm with the two rules improvement as Wu-Imp algorithm. All experimental results in this section are the averages of one thousand runs with the same N and R.

5.2. *Verification of DSPP-2*

In this subsection, we compare the results of optimal solution for shortest path connected dominating set (SPCDS) to the results produced by DSPP-2, Wu-Sim, and Wu-Imp algorithms. Since the SPCDS problem is NP-hard, we only check the optimal solution for small scale network topologies. We vary the total number of nodes N from 10 to 30 and the maximum transmission range R from 5 to 20. We randomly generate the topologies based on these two parameters, and calculate results on these topologies.

Table 1 shows the results for small scale networks. It shows the CDS size obtained by the optimal solution and the other three algorithms. From the optimal results, we find that when N increases, the optimal CDS size increases as well because we need more nodes in the set to guarantee that between any pair of nodes, there is a shortest path in the dominating set. On the other hand, when we fix N and increase R, the optimal CDS size decreases a little. The reason behind this is that

(1) With fixed node locations, increasing R will increase the number of edges, and a more strongly connected graph can be covered more cheaply. This explains when R increases, the optimal size decreases;
(2) With small transmission range, some topologies will not connected, and thus discarded by our topology generation procedure. The same topologies with a larger transmission range will be connected, and such type of topologies will require a large CDS. Such CDSs will make the average result larger. This is why the optimal size decreases only a little when R increases.

From Table 1, first of all, we can find that the output of DSPP-2 is closer to the optimal solution than Wu-Sim or Wu-Imp. We find that on average, the output of DSPP-2 is 0.01% larger than the optimal CDS, while the output of Wu-Imp is around 30% larger than the optimal solution and Wu-Sim gets about twice the

Table 1. Comparison among Wu, DSPP-2, and optimal solution.

		CDS size						CDS size			
N	R	Wu-Sim	Wu-Imp	DSPP-2	Optimal	N	R	Wu-Sim	Wu-Imp	DSPP-2	Optimal
10	5	6.59	4.85	3.72	3.71	10	15	6.42	4.53	3.48	3.46
15	5	10.44	8.19	6.29	6.29	15	15	11.03	8.01	5.88	5.88
20	5	15.32	11.86	8.88	8.87	20	15	15.57	11.43	8.46	8.41
25	5	19.97	15.26	11.3	11.28	25	15	19.86	14.99	11.35	11.33
30	5	24.96	19.13	14.51	14.47	30	15	24.69	18.15	13.49	13.45
10	10	6.7	4.85	3.78	3.77	10	20	6.49	4.63	3.45	3.44
15	10	10.95	8.29	6.14	6.12	15	20	10.97	7.19	5.71	5.66
20	10	15.51	11.52	8.69	8.63	20	20	15.89	11.06	7.99	7.97
25	10	20.04	15.06	11.07	11.06	25	20	20.68	14.82	10.76	10.7
30	10	24.82	18.9	14.27	14.21	30	20	25.11	18.47	13.69	13.63

number of nodes as the size of the optimal solution. Secondly, our two observations in the previous paragraph apply to Wu-Imp and DSPP-2 as well. However, when we keep N fixed and increase R, Wu-Sim does not show the same trend compared with the optimal solution. Therefore, DSPP-2 achieves results very close to the optimal solution, much better than the other two heuristic algorithms.

5.3. *CDS size*

In this subsection, we check one of the most important parameters for the algorithm, the CDS size. We vary the experimental setup from small scale topologies to large scale topologies. The number of nodes in the CDS determines the number of wireless devices which participate in the routing process, maintain routing tables, and exchange packets, which costs a large amount of energy. Thus this parameter is crucial for network performance. We choose N from 10 to 100 and we test four different maximum transmission ranges R: 15, 20, 25, and 30.

Figure 8 shows the results obtained by DSPP-2 as well as Wu-Sim and Wu-Imp. The CDS ratio is defined as the CDS size divided by the total number of nodes N. Next, we discuss several observations from these results.

Performance of our heuristic: Figure 8 shows that our DSPP-2 always achieves the lowest CDS ratio among these three algorithms. It means that fewer nodes will maintain the routing table when using our algorithm. More specifically, from the figures, we find that Wu-Sim always selects 65%–98% of the total nodes for the CDS. In other words, almost every node maintains the routing table. Wu-Imp achieves better solutions, in which about 45% of the nodes will be selected for the CDS when N is small, while around 70% of the nodes will be selected when N is large. Our DSPP-2 always gets the best results, and less than 50% of the nodes will be selected for the CDS even when the topology is large.

Trend discussion: All three algorithms show the same trend, as we discussed in Sec. 5.2. In detail, the CDS size increases when N increases. On the other hand,

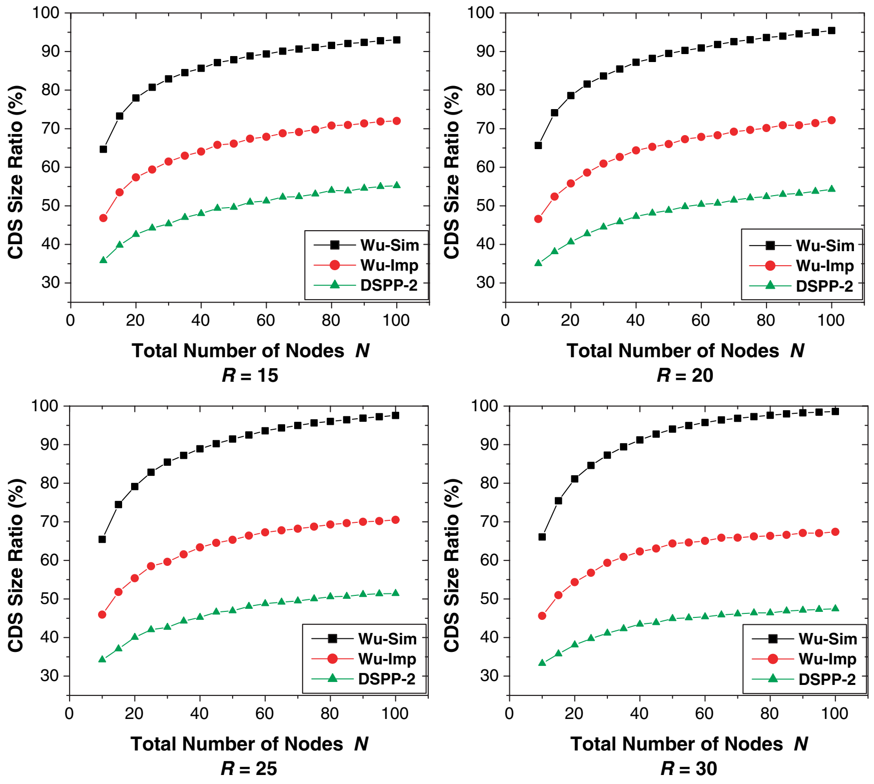


Fig. 8. (Color online) CDS size ratio versus number of nodes N.

when we keep N fixed, Wu-Imp and DSPP-2 show a slightly decreased CDS size as R increases, with the same reason as we discussed before. However, Wu-Sim exhibits an opposite trend, pushing more nodes into the CDS as R increases.

Explanation on trend variation: From Fig. 8, we can see that the CDS size ratio increases rapidly when N grows from 5 to 40. It still increases when N becomes larger, but becomes nearly constant. We can explain this trend as follows: when N is small, we do not have many choices to keep the topology connected. It is more likely that we will generate a graph with simple topology, where node pairs may have unique long shortest paths. Based on this, small size topologies will achieve small size CDS. When N increases, the topology becomes complicated, which means that each node has higher probability to be the intermediate node of some shortest paths. Thus the CDS size ratio increases rapidly. However, the increasing trend will slow down when N is sufficiently large. The reason is that when N increases, the topology has more chance to become a mesh, which allow diverse shortest paths between different node pairs. Thus each pair of nodes has more choices to connect with each other, and we do not need to insert all of these possibilities into our CDS size. Therefore, the CDS ratio almost keeps constant.

Finally, from the results, we can get the following conclusion: the total number of nodes N has greater effect on the CDS size than R does, which means that we can use a little bit smaller maximum transmission range (e.g. $R = 25$) and apply our DSPP-2 algorithm to achieve a good CDS size as well as save the power for the nodes in CDS.

5.4. *Diameter, Average Hop Count and Average Hop Distance*

In wireless networks, the key measurement to evaluate the system performance is energy consumption. A good topology control strategy can extend system lifetime and make it energy-efficient. In this subsection, we test another three main parameters which directly affect the power consumption in wireless networks: *Diameter* (DIA), *Average Hop Count* (AHC), and *Average Hop Distance* (AHD). *Diameter* denotes the longest Euclidean distance along a path among all shortest paths between pair of nodes in the network, and *Diameter Path* is this longest path. It measures the worst case of energy consumption when dealing with a unicast requirement. *Average Hop Count* is the average number of hops between any pair of nodes in the network. *Average hop distance* is the sum of the Euclidean distance for all pair shortest path divided by the number of pairs in the network. AHC together with AHD determine the average energy consumption for long term traffic load.

Simulation on diameter: Figure 9 shows the diameter evaluation. In Fig. 9(a), we first check the performance of DSPP-2. We want to measure the diameter divided by the maximum transmission range ($\frac{DIA}{R}$) — which denotes a minimum hop count for the *diameter path*. Through Fig. 9(a), we firstly get the hint that the smaller range R is, the more number of hops the diameter will go through, which means the longest path inside CDS will need more nodes to store and forward, leading to more energy consumption. Secondly, when we look at the point where $N = 100$, the diameter for $R = 15$ and 20 is around 120 while the diameter for $R = 25$ and 30

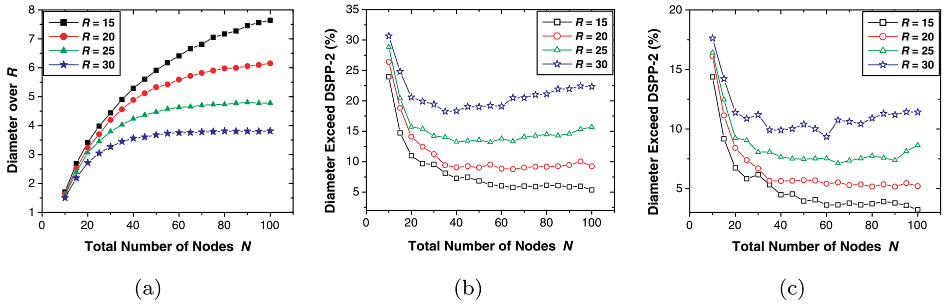
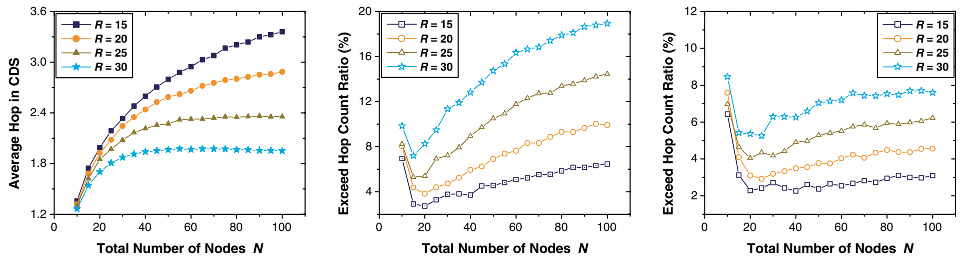


Fig. 9. (Color online) (a) Diameter evaluation for DSPP-2; (b) diameter difference ratio between Wu-Sim and DSPP-2; (c) diameter difference ratio between Wu-Imp and DSPP-2.

is around 100, which is smaller than 120. The reason behind this is that when the range is larger, more edges will introduced to the randomly generated topologies, which give us the opportunity to reduce the diameter. On the other hand, when N is small, the smaller range achieves smaller diameter because of the connectivity requirement for the topologies. Therefore, when the number of nodes is large, the network with larger transmission range performs better.

Figure 9(b) shows the percentage of difference between the Wu-Sim algorithm and DSPP-2 while Fig. 9(c) shows the difference between Wu-Imp algorithm and DSPP-2. The *Diameter Exceed Percentage* is defined as the diameter produced by Wu-Sim (for Fig. 9(b)) or Wu-Imp (for Fig. 9(c)) minus the diameter produced by DSPP-2 divided by the diameter of DSPP-2. From these two graphs, our DSPP-2 performs better with respect to diameter, which means that when the unicast requirement comes, DSPP-2 will consume less energy than the other two algorithms.

Simulation on average hop count: Figure 10 measures the average hop count. We first present the results produced by DSPP-2, shown in Fig. 10(a). From Fig. 10(a), when the range increases, the *average hop count* (AHC) decreases. Because if we increase R, each node can cover more area, which leads to fewer hop counts between node pairs. Furthermore, we see the connectivity requirement on the topologies leads to small AHC when N is small. When N is large, the AHC remains almost the same because the topology is more dense, which provides fewer hop counts for a given node pair. Therefore, although the number of node pairs increases, we can still get a low AHC. Same as Figs. 9(b) and 9(c), we test the other two algorithms and compare the results with those achieved by DSPP-2. The *Exceed Hop Count Ratio* is defined as average hop count achieved by Wu-Sim (for Fig. 10(b)) or by Wu-Imp (for Fig. 10(c)) minus that achieved by DSPP-2 over the average hop count achieved by DSPP-2. From these two figures, when a broadcast request comes, DSPP-2 always goes through fewer hops within the CDS than the other two algorithms, which means fewer number of intermediate nodes will store and forward the message.



(a) (b) (c)

Fig. 10. (Color online) Average hop count versus number of nodes: (a) average hop count evaluation for DSPP-2; (b) hop count difference ratio between Wu-Sim and DSPP-2; (c) hop count difference ratio between Wu-Imp and DSPP-2.

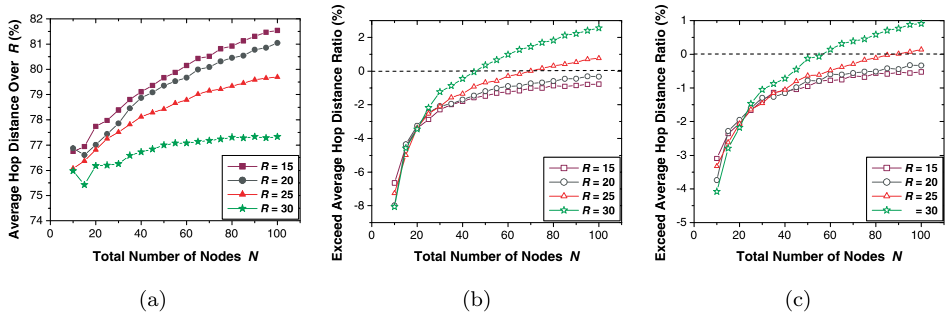


Fig. 11. (Color online) Average hop distance versus number of nodes: (a) average hop distance evaluation for DSPP-2; (b) diameter difference ratio between Wu-Sim & DSPP-2; (c) diameter difference ratio between Wu-Imp & DSPP-2.

Simulation on average hop distance: Figure 11 evaluates the last parameter: *average hop distance* (AHD). We first show the AHD produced by DSPP-2 in Fig. 11(a). In order to compare with R, we show a parameter named *Average Hop Distance Over* R, which is the average hop distance divided by the maximum transmission range ($\frac{AHD}{R}$). We find that when R increases, the *Average Hop Distance Over* R decreases. For example, we need more than 81% of R when $R = 15$, while we only need around 77% of R when $R = 30$, here $N \geq 80$. The reason behind this is that if R increases, node pairs have high probability to choose a shortest path with few hops.

Figures 11(b) and 11(c) show the average distance difference between Wu-Sim (for Fig. 11(b)) or Wu-Imp (for Fig. 11(c)) and DSPP-2. The interesting observation from these two figures is that both Wu-Sim and Wu-Imp achieve better AHD value than DSPP-2 in most cases, because these two algorithms put more nodes into the CDS. However, when we increase R and N, DSPP-2 can achieve a better solution than the other two algorithms.

In summary, from Figs. 9–11, we find that DSPP-2 always achieves the best performance for maximum energy consumption in the worst case and provides shortest paths with fewer hops for clients to forward messages. On the other hand, we find that Wu-Sim and Wu-Imp have better AHD value when N and R are small. When the topologies are dense, which means many nodes are located within a small area, DSPP-2 can also achieve better performance on AHD.

6. Conclusion and Future Work

Finding a *Connected Dominating Set* (CDS) is one key method for topology control. It is a strategy with energy-efficiency to reduce the cost of routing (e.g. reduce the routing table size) in wireless networks. In this paper, we study a variant of CDS with shortest path constraints to balance the size of CDS and client fairness. The goal is to find a minimum CDS such that the subset offers at least one shortest path for any two nodes, which has the same length compared to the shortest path of this pair in the original graph. We name this problem the *Shortest Path Connected Dominating Set* (SPCDS) problem.

We prove that SPCDS is NP-hard by reduction from *Hitting Set*, then provide theoretical analysis indicating that instead of finding a SPCDS, we can simply find a node subset which provides shortest path for all pairs of two hop neighbors from the original graph (SPP-2). Based on this analysis, we propose two efficient greedy heuristics for SPP-2 (one centralized and one distributed). We provide correctness, time complexity, message complexity, and approximation ratio analysis for these two algorithms and then evaluate their efficiency by comparison with the optimal solution in small scale networks, and comparison with Wu's work in [31], which demonstrates that the size of our solution is smaller. Our algorithm is the first distributed approach to select a CDS which guarantee the fairness for clients.

In the future, we may think about our communication model under *unit disk graph* to pursue a better approximation ratio. We may also think about algorithms for the k-spanner problem, where the length of path $P_{CDS}(u, v)$ through CDS is k times the length of path $P_V(u, v)$ in the original graph (Here $P_S(u, v)$ denotes the shortest path between u and v with all intermediate nodes belonging to vertex subset S).

7. Appendix A. Discussion on NP Issues for Shortest Path Connected Dominating Set Problem

To prove that SPCDS is NP-hard, we need to find a polynomial time reduction from an existing NP-hard optimization problem. We use the problem *Hitting Set* to make this reduction. Definition A.1 is the decision version of the *Hitting Set* problem.

Definition A.1 (Hitting Set). Given a finite set U with $|U| = m$, a family of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ where each $S_i \subseteq U$, and an integer K, a hitting set for $\mathcal{S}$ is a subset $H \subseteq U$ such that $H \cap S_i \neq \emptyset$, $\forall i = 1, 2, \ldots, n$, and $|H| \leq K$.

Theorem A.2. HS $\leq_m^p$ SPCDS.

Proof. As shown in Theorem 2.7, SPCDS problem is equivalent to SPP-2 problem, we need to prove HS $\leq_m^p$ SPP-2. Give an instance of *Hitting Set*. Now we try to build an instance of SPP-2. We make a graph G according to Algorithm 3.

Figure 12 is a simple example of this reduction. Here $U = \{u_1, u_2\}$, $S_1 = \{u_1, u_2\}$, and $\mathcal{S} = \{S_1\}$. We illustrate the topology by three steps separately (shown in Figs. 12(a)–12(c) respectively), and to make the graph easily recognized, in Fig. 12(c) we use dotted lines to connect (p_i, p_j) (This denotes the clique construction for all pairs in P).

It is trivial that Algorithm 3 takes polynomial time to finish the reduction. Next, we prove the correctness of this reduction.

Algorithm 3 Reduction from HS to SPP-2

```
    Input: A set $U$, and a family of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$.
    Output: An undirected graph $G = (V, E)$.
1: Begin
2:     Step 1: For each element $u_i \in U$, we create a node $u_i$, and for each subset $S_j$, we create two nodes $s_j^a$ and $s_j^b$. Set $V \leftarrow \{u_i\} \cup \{s_j^a\} \cup \{s_j^b\}$ $(i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n)$. For each $u_i \in S_j$, set $E \leftarrow E \cup \{(u_i, s_j^a), (u_i, s_j^b)\}$.
3:     Step 2: For every pair of nodes $(u, v)$ in $V$ except $(s_j^a, s_j^b)$, we create two nodes $p_k$, $q_k$, and three edges. Set $V \leftarrow V \cup \{p_k, q_k\}$ and $E \leftarrow E \cup \{(u, p_k), (v, p_k), (p_k, q_k)\}$.
4:     Step 3: For every pair of nodes $(p_i, p_j)$, set $E \leftarrow E \cup \{(p_i, p_j)\}$. So all $p_i$'s form a clique.
5: End
```

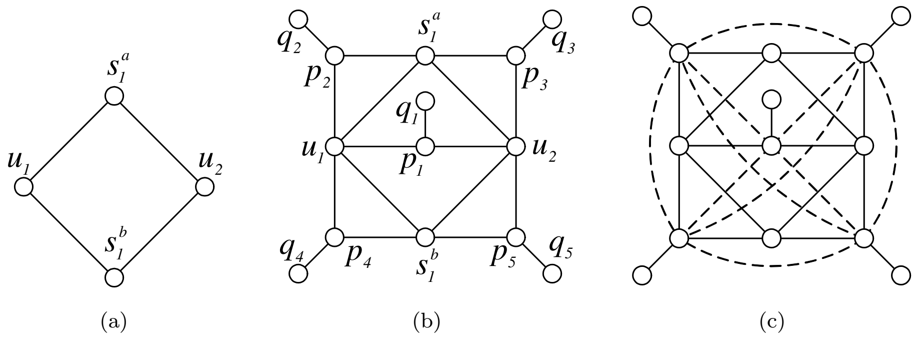


Fig. 12. Example for reduction. (a) result for Step 1; (b) result for Step 2; (c) result for Step 3.

HS $\Rightarrow$ SPP-2: Assume there exists a solution H for *Hitting Set* $(|H| \leq K)$. We are going to choose $P \cup H$ to form the result of SPP-2, where $P = \cup_k\{p_k\}$. Clearly $|P \cup H| \leq |P| + K$. Arbitrarily pick up two nodes (a, b) from G with distance 2, we prove that there exists a node $v \in P \cup H$ such that $(a, v) \in E$ and $(b, v) \in E$. We enumerate all possible conditions.

(1) If $a \in Q$ or $b \in Q$ $(Q = \cup_k\{q_k\})$, without loss of generality, let $a \in Q$. Then a only has one neighbor, p_k, which must belong to P. So a and b are connected by p_k, which is an element of $P \cup H$.
(2) If $a \in P$ or $b \in P$ (set $a \in P$ as well), since every node must have a neighbor in P, b has a neighbor p_k. Based on Algorithm 3, Step 3, the elements of P form a clique, which means $(p_k, a) \in E$.
(3) If $a, b \notin P \cup Q$, then $a, b \in \{u_i\} \cup \{s_j^a\} \cup \{s_j^b\}$.
    (a) If $(a, b) = (s_j^a, s_j^b)$ for some j, then by the correctness of the solution to hitting set, there exists some element $u_i \in H$ which belongs to set S_j. By the construction of Algorithm 3, $(s_j^a, u_i) \in E$ and $(s_j^b, u_i) \in E$.

(b) Otherwise, also by the construction of Algorithm 3, Step 2, there exists some p_k to connect a and b together, and $p_k \in P \cup H$.

Thus if there exists a solution of hitting set of size K or less, then there exists a solution of SPP-2 of size $|P| + K$ or less. We proved the first part.

HS $\Leftarrow$ SPP-2: Suppose we have a solution of SPP-2 of size $|P| + K$. First we need to prove that P is a subset of any feasible solution. Take an arbitrary element p_k of P, p_k has a neighbor q_k, and q_k's only neighbor is p_k. Therefore, p_k must belong to any feasible solution to connect q_k.

Next, take any pair of (s_j^a, s_j^b). They are not directly connected, and every path of length 2 goes through a node $u_i \in U$, where u_i is an element of S_j. By the correctness of the solution, one such u_i must belong to our solution. We combine them together to make a hitting set. Moreover, we know that the size of our solution is bounded by $|P| + K$, and as P must be in this solution, the size of such $\{u_i\}$ is bounded by K. Thus if there exists a solution of SPP-2 of size $|P| + K$ or less, then there exists a solution of hitting set of size K or less.

Therefore, we have that HS $\leq_m^p$ SPP-2, which proves Theorem A.2. □

References

[1] K. M. Alzoubi, P. J. Wan and O. Frieder, New distributed algorithm for connected dominating set in wireless ad hoc networks, in *Proc. 35th Hawaii Int. Conf. System Sciences*, Big Island, Hawaii (2002).

[2] K. M. Alzoubi, P. J. Wang and O. Frieder, Distributed heuristics for connected dominating sets in wireless ad hoc networks, *J. Commun. Netw.* **4** (2002) 22–29.

[3] C. Ambühl, T. Erlebach, M. Mihal'ák and M. Nunkesser, Constant-factor approximation for minimum weight (connected) dominating sets in unit disk graphs, in *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, in Proc. 9th Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2006) and 10th Int. Workshop on Randomization and Computation (RANDOM 2006) (Springer, Berlin/Heidelberg, 2006), pp. 3–14.

[4] B. S. Baker, Approximation algorithms for NP-complete problems on planar graphs, *J. ACM* **41** (1994) 153–180. Extended abstract published in the proceedings of FOCS'83 (1983), pp. 265–273.

[5] M. Burkhart, P. V. Rickenbach, R. Wattenhofer and A. Zollinger, Does topology control reduce interference? *Proc. 5th ACM Int. Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'04)* (2004), pp. 9–19.

[6] M. Cadei, X. Cheng and D. Z. Du, Connected domination in ad hoc wireless networks, in *Proc. 6th Int. Conf. Computer Science and Informatics* (2002).

[7] X. Cheng, X. Huang, D. Li, W. Wu and D. Z. Du, Polynomial-time approximation scheme for minimum connected dominating set in ad hoc wireless networks, *Networks* **42** (2003) 202–208.

[8] B. N. Clark, C. J. Colbourn and D. S. Johnson, Unit disk graphs, *Discrete Math.* **86** (1990) 165–177.

[9] U. Feige, A threshold of $\ln n$ for approximating set cover, *Proc. 28th ACM Symposium on Theory of Computing*, ACM (1996) 314–318.

[10] X. F. Gao, Y. C. Huang, Z. Zhang and W. L. Wu, $(6+\varepsilon)$-approximation for minimum weight dominating set in unit disk graphs, *The 14th Annual International Computing and Combinatorics Conference (COCOON 2008)*, Dalian, China, June 27–29 (2008).

[11] B. Gao, Y. H. Yang and H. Y. Ma, An efficient approximation scheme for minimum connected dominating set in wireless ad hoc networks, *IEEE Vehicular Technology Conference No. 60*, Los Angeles CA (2004).

[12] M. R. Garey and D. S. Johnson, *Computer and Intractablity: A Guide to the Theory of NP-Completeness* (Fressman, San Francisco, 1978).

[13] B. Han and W. J. Jia, Design and analysis of connected dominating set formation for topology control in wireless ad hoc networks, *Proc. 14th International Conference on Computer Communications and Networks (ICCCN 2005)* (2005), pp. 7–12.

[14] B. Han, H. H. Fu, L. D. Lin and W. J. Jia, Efficient construction of connected dominating set in wireless ad hoc networks, *Proc. First IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, Florida (2004).

[15] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz and R. E. Stearns, NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs, *J. Algorithms* **26** (1998) 238–274.

[16] B. H. Kim, J. K. Ahn, H. S. Kim, D. W. Roh, D. Y. Seo and S. H. Won, Packet transmission acknowledgement in wireless communication system, http://www.freepatentsonline.com/7372831.html.

[17] D. Y. Kim, W. Wang, C. C. Ma, N. Sohaee and W. L. Wu, A new clustering scheme for underwater acoustic sensor networks, submitted to *IEEE Transactions on Mobile Computing (IEEE TMC)*, 2008.

[18] K. Jain, J. Padhye, V. N. Padmanabhan and L. L. Liu, Impact of interference on multi-hop wireless network performance, *Wire. Netw.* **11** (2005) 471–487.

[19] Ö. Johansson, Simple distrbuted $\Delta+1$-coloring of graphs, *Information Processing Letters* **70** (1999) 229–232.

[20] Y. S. Li, M. T. Thai, F. Wang, C. W. Yi, P. J. Wan and D. Z. Du, On greedy construction of connected dominating sets in wireless networks, *Wire. Commun. Mobile Comput.* **5** (2005) 927–932.

[21] D. Lichtenstein, Planar formulae and their uses, *SIAM J. Comput.* **11** (1982) 329–343.

[22] C. Lund and M. Yannakakis, On the hardness of approximating minimization problems, *J. ACM* **41** (1994) 960–981.

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

[24] M. Min, H. W. Du, X. H. Jia, C. X. Huang, S. C. H. Huang and W. L. Wu, Improving construction for connected dominating set with steiner tree in wireless sensor networks, *J. Global Optim.* **35** (2006) 111–119.

[25] How Does Wireless Internet Work, http://www.superpages.com/supertips/how-does-wireless-internet-work.html.

[26] N. B. Salem and J. P. Hubaux, A fair scheduling for wireless mesh networks, *The 1st IEEE Workshop on Wireless Mesh Networks*, Held in conjunction with SECON'05, Santa Clara (2005).

[27] M. T. Thai and D. Z. Du, Connected dominating sets in disk graphs with bidirectional links, *IEEE Commun. Lett.* **10** (2006) 138–140.

[28] M. T. Thai, F. Wang, D. Liu, S. W. Zhu and D. Z. Du, Connected dominating sets in wireless networks with different transmission ranges, *IEEE Trans. Mobile Comput.* **6** (2007) 721–730.

[29] P. J. Wan, K. M. Alzoubi and O. Frieder, Distributed construction of connected dominating set in wireless ad hoc networks, *Proc. Third ACM Internet, Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications* (1999), pp. 7–14.

[30] Y. Wang and X. Y. Li, Localized construction of bounded degree and planar spanner for wireless ad hoc networks, *Mobile Netw. Appl.* **11** (2006) 161–175.

[31] J. Wu and H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, *Proc. 3rd ACM Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications* (1999), pp. 7–14.

[32] Z. Zhang, X. F. Gao, W. L. Wu and D. Z. Du, PTAS for minimum connected dominating set in unit ball graph, *The 3rd International Conference on Wireless Algorithms, Systems and Applications (WASA 2008)* Oct. 26–28 (2008).