A Novel Approximation for Multi-Hop Connected Clustering Problem in Wireless Networks

Xiaofeng Gao, *Member, IEEE*, Xudong Zhu, Jun Li, Fan Wu, *Member, IEEE*, Guihai Chen, *Member, IEEE*, Ding-Zhu Du, *Member, IEEE*, and Shaojie Tang, *Member, IEEE*

***Abstract*— Wireless sensor networks (WSNs) have been widely used in a plenty of applications. To achieve higher efficiency for data collection, WSNs are often partitioned into several disjointed clusters, each with a representative cluster head in charge of the data gathering and routing process. Such a partition is balanced and effective, if the distance between each node and its cluster head can be bounded within a constant number of hops, and any two cluster heads are connected. Finding such a cluster partition with minimum number of clusters and connectors between cluster heads is defined as *minimum connected d-hop dominating set* (d-MCDS) problem, which is proved to be NP-complete. In this paper, we propose a distributed approximation named *CS-Cluster* to address the d-MCDS problem under *unit disk graph*. CS-Cluster constructs a sparser d-hop maximal independent set (d-MIS), connects the d-MIS, and finally checks and removes redundant nodes. We prove the approximation ratio of CS-Cluster is $(2d+1)\lambda$, where λ is a parameter related with d but is no more than 18.4. Compared with the previous best result $O(d^2)$, our approximation ratio is a great improvement. Our evaluation results demonstrate the outstanding performance of our algorithm compared with previous works.**

***Index Terms*— Wireless networks, clustering, distributed algorithm, approximation, connected dominating set.**

I. Introduction

WIRELESS sensor network (WSN) is a self-organized communication system consisting of many small-sized, cheap, and battery-powered sensors. WSNs have been widely used in a lot of applications such as health-care industry, food industry, disaster management, battlefield surveillance, etc. In these applications, the task of each sensor is to collect the information in its surrounding environment and transmit the corresponding data to the base stations of WSNs. Over the past decades, a lot of related researches have studied data gathering and transmission problems [1]–[7].

Due to the battery limitation of sensors, energy conservation is always an important factor for extending the lifetime of WSNs, and a lot of researches have been done for this problem [6], [8]–[11]. In addition, sensor nodes also have constraints on communication bandwidth, communication range, and storage space. Thus, a message may be transferred multiple times through several intermediate nodes along a path to its destination. This kind of flooding-like routing scheme causes huge amount of traffic collision, message redundancy, and energy consumption.

In order to overcome these shortcomings, an efficient approach named clustering has been widely used by many researchers. We can divide a WSN into several disjointed clusters, each with a cluster head to take charge of the data gathering and the communication processes. Then, any node in the WSN only needs to collect information and send the data to its corresponding cluster head, which saves a lot of energy. The most important part for clusting scheme is to efficiently partition the given WSN into disjointed clusters and many algorithms were proposed to achieve this purpose. In [12], through coordination of nodes in the same cluster, the authors proposed a clustering algorithm to optimize the energy conservation. Load balancing is also a crucial issue for clustering mechanisms. Younis and Fahmy [13] aimed at setting equal-sized clusters to balance the workload. By setting a hop threshold k, researchers also discussed k-hop clustering problem [14]–[16] requiring that each node should be within k hops from its cluster head.

The average size of clusters is an important metric to estimate the performance of a WSN. If the cluster size is too large, then it is difficult for the cluster head to manage the cluster. In contrast, if the cluster size is too small, then there will be too many clusters in the WSN, which downgrades the performance of clustering strategy. An effective way to control the average size is to set a parameter d, and each cluster head only takes charge of its d-hop neighbors. To achieve better communication between clusters, any pair of cluster heads should be able to communicate with each other directly or through the relay of other cluster heads. Thus the problem becomes finding an effective partition with minimum number of clusters w.r.t. d, while each cluster head can generate a connected subgraph with the smallest number of additional connectors.

More precisely, we use a graph $G = (V, E)$ to model a WSN, where V is the set of sensors in the network while $(u, v) \in E$ iff u and v communicate with each other. In this

Manuscript received March 30, 2016; revised December 8, 2016; accepted March 1, 2017; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Chong. Date of publication May 2, 2017; date of current version August 16, 2017. This work was supported in part by the Program of International S&T Cooperation under Grant 2016YFE0100300, in part by the China 973 Project under Grant 2014CB340303, in part by the National Natural Science Foundation of China under Grant 61472252, Grant 61672353, Grant 61422208, and Grant 61672348, in part by the Shanghai Science and Technology Fund under Grant 15220721300, and in part by the Scientific Research Foundation for the Returned Overseas Chinese Scholars. *(Corresponding Author: Xiaofeng Gao.)*

X. Gao, X. Zhu, J. Li, F. Wu, and G. Chen are with the Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: gao-xf@cs.sjtu.edu.cn; xudongzhu42@gmail.com; lijun2009@sjtu.edu.cn; fwu@cs.sjtu.edu.cn; gchen@cs.sjtu.edu.cn).

D.-Z. Du and S. Tang are with The University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: dzdu@utdallas.edu; tangshaojie@gmail.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This consists of a PDF (271 KB) containing the appendices.

Digital Object Identifier 10.1109/TNET.2017.2690359

paper, we consider a homogeneous network, where each node has the same communication range. So the graph can be formed as a *unit disk graph* (UDG). We focus on constructing a connected d-hop clusters for G. Actually, the set of cluster heads can be considered as a d-hop dominating set (d-DS). Consequently, our task is to find a connected d-DS (d-CDS) for a given graph. Moreover, we hope the cardinality of the d-CDS is minimized in order to reduce the maximum possible number of redundant messages, which corresponds to the *minimum connected d-hop dominating set* (d-MCDS) problem.

In all, our goal is to find a d-MCDS for a UDG $G = (V, E)$. In this paper, we propose a three-phase algorithm named *Connected Sparse Clustering Strategy* (CS-Cluster) to solve this problem. The first phase of CS-Cluster is to select a d-hop maximal independent set (d-MIS). Then, some extra nodes are added to connect the d-MIS into a d-CDS. In order to further reduce the size of the obtained d-CDS, the third phase is to check and remove redundant nodes. Our contributions in this paper are summarized as follows:

- We propose a novel approximation algorithm for the d-MCDS problem, and prove its approximation ratio as $(2d + 1)\lambda$, where λ is a parameter related with d but no more than 18.4. Our algorithm improves the previous best ratio of $O(d^2)$ into $O(d)$. Moreover, evaluation results also exhibit the outstanding performance of CS-Cluster compared with the closest related work.
- We theoretically analyze the upper bound of the size of d-MIS (denoted as $\alpha(G)$) w.r.t. the size of d-MCDS (denoted as $\gamma(G)$) for a UDG $G = (V, E)$. We prove that the ratio of $\alpha(G)$ and $\gamma(G)$ is bounded by λ. Compared with the previous best result $O(d)$, we reduce it to $O(1)$, which is a huge improvement.
- CS-Cluster is a distributed algorithm, which is more suitable for applications in WSNs. With thorough discussions, we conclude that CS-Cluster not only fits for UDG model, but also for general network model.

Our paper is organized as follows. Section II provides some preliminaries. In Section III, we introduce the related works. In Section V, we describe our algorithm CS-Cluster for the d-MCDS problem. Afterwards, we analyze the performance of CS-Cluster in Section VI and Section VII gives the simulation. Finally, Section VIII concludes this paper.

II. Preliminaries

In this section, we will introduce some definitions and then summarize the symbols used in the paper. Firstly, for a given graph $G = (V, E)$, we give several definitions.

*Definition 1 (UDG): G is a **Unit Disk Graph (UDG)** if $\forall u, v \in V$, there is an edge $(u, v) \in E$ iff $dis(u, v) \leq 1$. $dis(u, v)$ is the Euclidean distance between u and v.*

*Definition 2 (d-DS): $D \subseteq V$ is a **d-hop Dominating Set (d-DS)** of G if $\forall v \in V$, either $v \in D$ or $\exists u \in D$ such that there exists a path of length at most d-hop between u and v.*

*Definition 3 (d-IS): A **d-hop Independent Set (d-IS)** of G is a subset $I \subseteq V$ such that $\forall u, v \in I$, there does not exist a path of length at most d-hop between u and v.*

*Definition 4 (d-MIS): A d-IS I is a **d-hop Maximal Independent Set (d-MIS)** if $\forall v \in V \backslash I$, $I \cup \{v\}$ is no longer a d-IS.*

TABLE I
Symbols, Notations, and Functions in This Paper

Notation	Explanation
$dis(u, v)$	the Euclidean distance between u and v.
$\|S\|$	the cardinality of set S.
$N^r(v)$	the r-hop neighbor set of v with nodes which are at most r-hop away from v.
$N^r(S)$	the r-hop neighbor set of S with nodes which are at most r-hop away from some node in S.
$disk_r(v)$	the disk with center v and radius r.
$A_r(v)$	the area of $disk_r(v)$.
$A_r(S)$	the area of $\cup_{v \in S} disk_r(v)$
$Area(P)$	the area of geometrical shape P.

Many researchers consider finding a d-MIS instead of a d-DS for graph G, since the former is easier and more efficient. Thus we need the following lemma.

Lemma 1: For any given graph G, a d-MIS is also a d-DS.

Proof: For a d-MIS I and any node $v \in V \backslash I$, $I \cup \{v\}$ is no longer a d-MIS, which means v is dominated by some node in I in d hops. As a result, I is a d-DS. □

*Definition 5 (d-CDS): A subset $C \subseteq V$ is a **Connected d-hop Dominating Set (d-CDS)** of G if C is a d-DS and the subgraph induced by C is connected.*

In addition, we need to define the Voronoi Division which will be referred in next sections.

Definition 6 (Voronoi Division): Let S be a set of nodes in Euclidean space. For each node $v \in S$, the corresponding Voronoi cell $V(v)$ is the set of points that are closer to v than to other nodes of S, which means

$$V(v) = \{w \mid \textit{for every } u \in S \backslash \{v\}, dis(v, w) \leq dis(u, w)\}.$$

The Voronoi diagram is the partition induced by Voronoi cells.

In order to simplify our problem, we make some assumptions. First, we focus on WSNs located at 2-dimensional space. Second, each sensor in the network has the same communication range and implements efficient scheduling strategy with multiple available frequencies, such that no collision occurs in the procedure of message transmitting. Under these assumptions, we define the d-MCDS problem as follows.

Definition 7 (d-MCDS): For a given UDG $G = (V, E)$, d-MCDS problem is to find a d-CDS with minimum size.

Table I introduces and summarizes the symbols, functions, and notations that will be used in the following sections.

III. Related Works

The minimum connected dominating set problem (namely, the 1-MCDS problem) is a classical NP-complete problem. In [17], Clark et al. first proved that MCDS is NP-complete even in UDG. To obtain a better feasible solution, Wan *et al.* [18] devised a two-phase algorithm with constant-factor approximation ratio. The first phase is to select an MIS. Then, it adds some extra nodes to connect the MIS into a CDS. Later, many works were done to improve this approximation ratio [19]–[21]. The ratio of the size of MIS to the size of MCDS in graph G is crucial to estimate the algorithm's performance. The upper bound of such ratio is also called the theoretical bound to approximation CDS. Up to now, the best

result for this bound is 3.399 [22]. Kim *et al.* [23] provided an approximation for MCDS in unit ball graph (UDG).

The d-MCDS problem is an extension of 1-MCDS. In 2000, Vuong and Huynh [24] proved that d-MCDS is NP-complete in general graph by a reduction from 3-SAT problem. Later, Nguyen and Huynh [25] proved its NP-completeness in UDG. A lot of heurestics were proposed to find a feasible solution of d-MCDS [26], [27]. However, they all lacked approximation analysis.

In 2010, Li and Zhang [28] proposed an approximation algorithm on finding minimum two-connected d-hop dominating set. They gave an approximation ratio of $O(\log |V|)$. Later, Gao *et al.* [29] presented a two-phase distributed algorithm to compute d-CDS in UDG, and they gave a constant-factor approximation ratio of $O(d^3)$. Zhang *et al.* [30] improved the approximation ratio into $O(d^2)$. In 2014, Zhu *et al.* [31] proposed the first constant-factor approximation for d-MCDS in 3-dimensional space.

There are many other related works for the clustering problem. Wang *et al.* [32] proposed a PTAS to minimize the average hop distance from any nodes to cluster heads in 2D sensor networks without connectivity property. Kim *et al.* [33] discussed how to find the locations of k sinks such that the maximum distance from other nodes to sinks is minimized. Some researches aimed at fault-tolerant virtual backbone construction [34]–[36], and formulated this problem as m-dominating connected set. In [37] and [38], the authors consider latency of the network and try to find connected dominating set with bounded diameter.

IV. CS-Cluster for the d-MCDS Problem

In this section, we introduce our algorithm named *Connected Sparse Clustering Strategy* (CS-Cluster) for the d-MCDS problem in $G = (V, E)$. CS-Cluster consists of three phases. First, we construct a sparse d-MIS. Second, we connect the d-MIS into a d-CDS by adding some extra nodes called connectors. Finally, we remove redundant nodes from the obtained d-CDS to further reduce its size.

CS-Cluster is a coloring algorithm. We use four different colors to denote different statuses. Initially, all nodes are white. When a node is chosen as a dominator, it will be colored as black. Once a node has a black neighbor within d hops, which means it is dominated by some black node, it will be colored grey. Nodes used to connect dominators are colored blue.

In addition, when applying CS-Cluster to partition a wireless network, we select those black nodes as cluster headers. For each cluster head, its corresponding cluster consists of its neighbors which are at most d hops away.

A. Constructing a d-MIS

According Lemma 1, we can select a d-MIS as d-DS for a given graph. Generally, we select d-MIS nodes one by one. For example, assume the current d-IS is S, then we select a node from $V \setminus \cup_{v \in S} N^d(v)$ and add it to S. We continue this process until S is a d-MIS. In literature such as [29]–[31], researchers often restrict that the new node is $d+1$ hops away from S. With such restriction, they can easily use spanning tree algorithm to connect the final d-MIS.

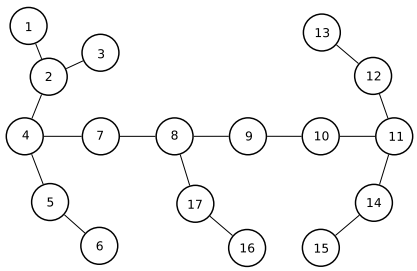


Fig. 1. The input graph of a 2-MCDS problem with 17 nodes.

However, such restriction has a great effect on the size of the chosen d-MIS. An example is shown in Fig. 1 with $d = 2$. Considering the restriction, we will select nodes $\{4, 9, 12, 15, 16\}$ as d-MIS with size 5. However, if we ignore such restriction, we can select a set $\{4, 11, 16\}$ with size 3. Based on this observation, we remove such restriction, and propose an efficient algorithm to select a sparser d-MIS as Alg. 1 shows.

Algorithm 1 Constructing d-MIS

Input : A connected graph $G = (V, E)$
Output: A d-MIS of G

```
Color all nodes in V as white;
M = ∅;
while there exists a white node in V do
    Find a white node v from V\M, such that the number
    of v's white neighbors within d hops is maximum.
    Use id to break ties;
    Color v as black and color its white neighbors within
    d hops as grey;
    M = M ∪ {v};
return M.
```

Alg. 1 is a greedy algorithm. Initially, it color all nodes in V as white. Then, for each white node v, it calculates the number of v' white neighbors within d hops, which is called as v's degree. Among all white nodes, we select the node with maximal degree, and color it black. Meanwhile, update the statuses, such as colors and degrees, for white nodes within d hops away from the new chosen node. Next, continue to choose the node with maximal degree from the remaining white nodes, until no white node exists.

Fig. 2 shows the corresponding 2-MIS after processing Alg. 1 for the 2-MCDS problem in Fig. 1.

B. Connecting a d-MIS

After Alg. 1, we get a d-MIS denoted as M, which is also a d-DS. Now, we discuss how to connect it into a d-CDS.

For a given graph $G = (V, E)$, any two nodes $u, v \in V$ are "k-hop connected" to each other if there exists a path in graph G between them and the length of the path is at most k. For example, in Fig 2, node 4 and node 16 are 4-hop connected and also 5-hop connected. A subset $S \subseteq V$ is a "k-hop connected d-hop dominating set" iff:

- S is a d-hop dominating set of graph G.

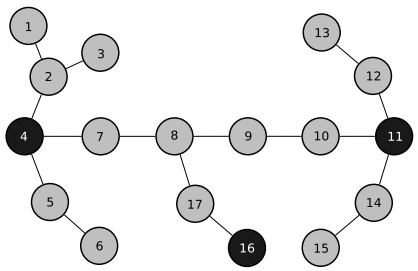


Fig. 2. The result after processing Alg. 1. Initially, node 4 has the maximal white neighbors with 2 hops. Thus, we color node 4 black and all its white neighbors with 2 hops {1, 2, 3, 5, 6, 7, 8} grey. In the next turn, node 11 with 6 white neighbors with 2 hops is chosen. Thus, color node 11 black and its white neighbors with 2 hops {9, 10, 12, 13, 14, 15} grey. In the third iteration, both node 16 and node 17 have one white neighbor within 2 hops. We break tie by selecting the node with smaller id. Thus, color node 16 black and color node 17 gray. Since there are not more white nodes, Alg. 1 terminates.

- For any vertex $u \in S$, there exists at least one vertex $v \in S$ such that u and v are k-hop connected.

For example, in Fig 2, the set of nodes $\{4, 11, 16\}$ forms a 5-hop connected 2-hop dominating set. In addition, for a graph $G = (V, E)$, a subset $S \subseteq V$ is a "k-hop connected component" iff:

- For any vertex $u \in S$, there exist at least one vertex $v \in S$ such that u and v are k-hop connected.
- For any other vertex $v \in V \backslash S$, there does not exist any vertex $u \in S$ such that u and v are k-hop connected.

For example, in Fig 2, nodes 4, 8, 16 form a 2-hop connected component. For nodes $\{4, 8, 11, 16\}$, they form a 3-hop connected component.

For a vertex subset $C \subseteq V$, let $f_k(C)$ be the number of k-hop connected components of the induced subgraph $G[C]$. For example, in Fig. 2, $f_2(\{4, 8, 16\}) = 1$, $f_2(\{4, 8, 11, 16\}) = 2$. $\forall v \in V$, define

$$-\Delta_v f_k(C) = f_k(C) - f_k\left(C \cup \{v\}\right),$$

representing the reduced number of k-hop connected components in C when v is inserted into C.

1) Algorithm Description: The main idea is divide and conquer and the algorithm consists of several subprocedures. In each subprocedure, we add some extra nodes to make M more closer than the previous subprocedure.

In detail, in the i^{th} subprocedure, a subset C_i is selected from $V \backslash \bigcup_{j<i} C_i \cup M$ such that $\bigcup_{j \le i} C_j \cup M$ is a $r(i)$-hop connected d-hop dominating set. The definition of $r(i)$ is shown as follows:

$$\begin{cases} r(0) = 2d + 1, \\ r(i) = \left\lfloor \frac{r(i-1)+1}{2} \right\rfloor. \end{cases} \tag{1}$$

Obviously, nodes in $r(i)$-hop connected d-hop dominating set is more closer than in $r(i-1)$-hop connected d-hop dominating set, since $r(i)$ is a monotone decreasing function.

After Alg. 1, M is an $r(0)$-hop connected d-hop dominating set, which means any two nodes in M are $(2d + 1)$-hop connected. When the algorithm in this subsection terminates, we should get a 1-hop connected d-dominating set, namely d-CDS. Alg. 2 depicts the detailed steps.

Algorithm 2 Connecting d-MIS

Input : $G = (V, E)$, a d-MIS M of G
Output: A d-CDS of G

```
C = M, r = 2d + 1, i = 0;
while r > 1 do
    r = ⌊(r+1)/2⌋, i = i + 1, C_i = ∅;
    while f_r(C ∪ C_i) > 1 do
        Find a grey node v from V\(C ∪ C_i) with the
        maximum cost(v). Use id to break ties;
        Color v as blue;
        C_i = C_i ∪ {v};
    C = C ∪ C_i;
return C.
```

Hence, the key part in Alg. 2 is how to determine nodes in C_i. To make the size of C_i as small as possible, we should choose the most efficient nodes that makes $\bigcup_{j \le i} C_j \cup M$ form an $r(i)$-hop connected d-hop dominating set. An intuitive idea is to iteratively choose nodes that can reduce the most number of $r(i)$-hop connected components. In order to make our algorithm more economical, we also take the number of the fewest nodes to reduce those $r(i)$-hop connected components into account. The detailed description is as follows.

At the beginning of the t^{th} iteration of the i^{th} round, C_i contains $t-1$ nodes. Let $C = \bigcup_{j \le i} C_i \cup M$. For a node v in $V \backslash C$, it reduces the number of $r(i)$-hop connected components by $-\Delta_v f_{r(i)}(C)$. Consequently, there exist $-\Delta_v f_{r(i)}(C)$ shortest paths that connect node v to those $-\Delta_v f_{r(i)}(C)$ components respectively. We use $no.(v)$ to denote the total number of nodes in those shortest pathes except two end points (but includes v). That is to say, $no.(v)$ is the minimum number of nodes with respect to v to reduce those $-\Delta_v f_{r(i)}(C)$ components. Moreover, we use $cost(v)$ to denote the cost of v under the current state and its definition is as follows.

$$cost(v) = \frac{-\Delta_v f_r(C \cup C_i)}{no.(v)}.$$

In every iteration, we select the node with the largest cost. Note that the definition of $cost(v)$ is much crucial to guarantee and improve the performance of our algorithm. An example of Alg. 2 is shown in Fig. 3.

2) Correctness Proof: Initially, the input M is a d-MIS, so $f_{d+1}(M) = |M|$ and $f_{2d+1}(M) \le |M|$. According to the definition of d-MIS, the nodes in d-MIS can be ordered in a way such that each node is at most $(2d+1)$-hop away from one of its predecessors. We call such a property as "$(2d + 1)$-hop connection property". Hence, easy to see that $f_{2d+1}(M) = 1$. Actually we have Lemma 2 as follows.

Lemma 2: For any subset S, it has k-hop connection property if and only if $f_k(S) = 1$.

Proof: It is clear that S has k-hop connection property if and only if the subgraph induced by S is k-hop connected, which is true when $f_k(S) = 1$. □

Lemma 3: In Alg. 2, the i^{th} round will terminate when $f_{r(i)}(\bigcup_{j \le i} C_j \cup M) = 1$.

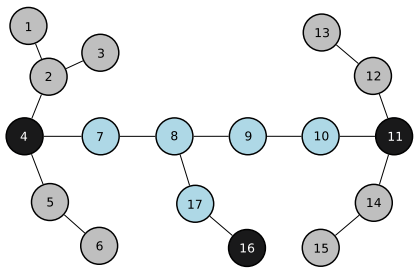


Fig. 3. The result after processing Alg. 2. At the beginning of the first round, there are three 3-hop connected components ({4}, {11}, {16}). We should 3-hop connect these three components into one. At this time, the costs of nodes 7, 8, 9, 17 are 1/3, 2/5, 2/5, 1/3 respectively, while all other grey nodes have cost 0. Since node 8 has smaller id than node 9, we color node 8 blue. Now the 3-hop connected component is only {4, 8, 11, 16}, and thus the first round terminates. Next, in the second round there are two 2-hop connected components ({4, 8, 16}, {11}). At this time nodes 9, 10 have the same cost of 1/2 while other grey nodes have value 0. Similarly we color node 9 blue and terminate the second round. When the third round starts, there are four 1-hop connected components ({4}, {8, 9}, {11}, {16}), while nodes 7, 17, 10 have the same cost of 1 and other grey nodes have cost 0. Thus, we select node 7 in the first iteration of this round. Then, select 10 in the second iteration, and 17 in the third iteration. After that, Round 3 terminates. At this time $r = 1$. Consequently, Alg. 2 completes its task.

Proof: When $i = 0$, Lemma 3 holds obviously as we have shown above. Assume Lemma 3 holds for the i^{th} round where $i \geq 0$, we show that it also holds for the $(i+1)^{th}$ round.

Obviously, the $(i+1)^{th}$ round starts with $f_{r(i)}(\bigcup_{j\leq i} C_j \cup M) = 1$. In this round, we need to connect all those $r(i+1)$-hop connected components into a whole $r(i+1)$-hop connected component. It is clear that as long as $f_{r(i+1)}(\bigcup_{j\leq i+1} C_j \cup M) > 1$, there must exist more than one $r(i+1)$-hop connected components. And these $r(i+1)$-hop connected components are $r(i)$-hop connected because the set $\bigcup_{j\leq i} C_j \cup M$ has the $r(i)$-hop connection property. Thus, there must exist two $r(i+1)$-hop connected components which are $r(i)$-hop connected, and a path of length at most $r(i)$ exists between them. The middle node in this path is at most $\lfloor \frac{r(i)+1}{2} \rfloor = r(i+1)$ away from these two components. So it can connect these components into one.

Thus, at each iteration in the $(i+1)^{th}$ round, the algorithm can always find a satiable node that can reduce the number of $r(i+1)$-hop connected components by at least 1. Therefore, Alg. 2 will continually execute until the round reaches the final state $f_{r(i+1)}(\bigcup_{j\leq i+1} C_j \cup M) = 1$. Then Lemma 3 holds. □

Theorem 1: Alg. 2 connects a d-MIS into a d-CDS.

Proof: According to Lemma 3, Alg. 2 ends up at $f_1(\bigcup_j C_j \cup M) = 1$, which means the subgraph induced by $\bigcup_j C_j \cup M$ is connected. On the other side, set M alone can dominate V. Thus, $\bigcup_j C_j \cup M$ is a d-CDS. □

C. Remove Redundant Nodes

After the first two phases, we obtain a d-CDS and denote it as C. And all nodes in C are either black or blue. It is not difficult to find that there may exist numbers of redundant nodes in C. That is to say, after removing those redundant nodes, the remaining nodes can still form a d-CDS. In Fig.3, it is easy to figure out that node 16 and node 17 are redundant. In this section, we will discuss how to further reduce the size of C by checking and removing the redundant nodes in C.

From the perspective of the d-MCDS problem, a node v in d-CDS is redundant iff:

- Every node that v dominates must have at least one alternative dominator.
- The subgraph induced by $C - \{v\}$ is connected.

The first requirement is to guarantee the property of domination. That is to say, after we remove those redundant nodes, the remaining nodes in C can still dominate the whole network within d hops. In this requirement, the dominator refers to black node or blue node.

The second requirement can be analyzed in details. For any node $v \in C$, there are two situations to discuss. First, in the subgraph $G[C]$ induced by C, if v is a leaf node, namely v's degree in $G[C]$ is one, then removing v has no effects on the connectivity of the subgraph $G[C-\{v\}]$ induced by $C-\{v\}$. Secondly, if v is connected by more than one connectors, namely v's degree in $G[C]$ is more than 1, then the subgraph $G[C-\{v\}]$ is connected only when these connectors are connected. To check whether those connectors are connected, it may involve the whole subgraph $G[C-\{v\}]$. For example, if all nodes in C form a ring, v's connectors can be connected by all the other nodes in $G[C-\{v\}]$. In this sense, the time complexity is huge.

Obviously, if we want to find all the redundant nodes from C, it will cost a lot of time. Hence, we should make trade-off between the time complexity and the final performance. In this paper, we only consider the first situation above to effectively control the time complexity of this phase. In this situation, we only need to check the first requirement to decide whether v is redundant.

Actually, the trade-off we made is reasonable. In the first phase, we make the distance between adjacent independent nodes as large as possible. Consequently, there are a lot of independent nodes in C located on the boundary area of the whole wireless network. On the other side, it is easy to observe that when a dominator lies in the boundary area of a homogenous wireless network, its coverage area includes a large area that are outside the network. This is actually a kind of waste. From this point, we can conclude that, if we could move those boundary dominators towards the inner of the network, we can make better use of those nodes. Besides, there is also a bigger opportunity for these boundary dominator to be a redundant node than inner nodes.

Based on the discission above, we can further reduce the size of C by Alg. 3. In each iteration, we only consider black nodes which are leaves in the subgraph of $G[C]$. For any black leaf node in C, if all its dominatees have alternative dominators, we remove it from C and update its color as grey. Afterwards, if v's direct connector is now also a leaf in C, we change this connector's color as black. Alg. 3 continues to find such redundant black nodes until such nodes don't exist.

Fig. 4 is an example to illustrate the performance of Alg. 3 after processing Alg. 1 and Alg. 2 for Fig. 1.

V. Distributed CS-Cluster

In this section, we will introduce a distributed algorithm, namely Distributed CS-Cluster, for the d-MCDS problem.

Algorithm 3 Removing Redundant Dominators

Input : $G = (V, E)$ and a d-CDS C of G
Output: A smaller d-CDS of G

$L \leftarrow \{v|\ v$ is a black leaf node in $G[C]\}$;
while *there exists a redundant node* $v \in L$ **do**
  color v grey, remove v from C;
  if *v's direct connector u is a leaf in $G[C]$* **then**
    color u black, add u into L;
return C.

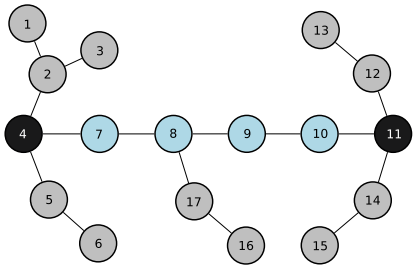


Fig. 4. The result after processing Alg. 3. At the beginning of Alg. 3, there are three black nodes, nodes 4, 11, 16 and $L = \{4, 11, 16\}$. By checking, we find every node (nodes 8, 17) dominated by node 16 has alternative dominators (themselves). Thus, node 16 is redundant. Remove it from C and color it grey. Moreover, the degree of node 17 in $G[C]$ at this time is one. Thus, color node 17 black and add it to L. In the second iteration, we have three black nodes, nodes 4, 17, 11 and $L = \{4, 11, 17\}$. Similarly, we find that node 17 is redundant and remove it. Since the blue neighbor of node 17 is node 8 and the degree of node 8 is not 1, we do not change the color of node 8. Consequently, we have two black nodes, nodes 4, 11 and $L = \{4, 11\}$. By checking, neither of them is redundant. Thus, Alg. 3 terminates.

Similarly with CS-Cluster, the Distributed CS-Cluster also consists of three phases.

For each node u, the local variables it contains are listed as follows.

- $id(u)$: u's unique identification.
- $color(u)$: u's color.
- $degree(u)$: The number of u's white neighbors within d hops.
- $neighbor\_info(u)$: The information of nodes which are at most d hops away from u.
- $neighbor\_2d\_info(u)$: The information of nodes whose hop distances to node u are from d to $2d$.
- $componentNum(u)$: The number of component u belongs to. Initially, the value of $componentNum(u)$ is equal to $id(u)$.
- $connectorCost(u)$: A collection of costs of nodes which can connect u with other components

For $neighbor\_info(u)$, each item in it is a collection, whose structure is $[id, color, degree, dis]$, and corresponds with one of u's neighbor within $2d$ hops. For item $elem$, which corresponds to node v, $elem.id$, $elem.color$ and $elem.degree$ refer to v's id, v's color, and v's $degree$ respectively. In addition, $elem.dis$ means the the hop distance between v and u. $elem.degree$. Besides, $neighbor\_2d\_info(u)$ is analogous to $neighbor\_info(u)$.

A. Construct d-MIS

Alg. 4 shows the construction of a d-MIS distributedly. The main idea is similar with Alg. 1 and we can select a d-MIS iteratively based on the $degree$ of each white node.

Messages involved in this phase are listed as follows.

- HELLO: Each node send this message to tell its existence to its neighbors.
- BLACK: Once a node is colored black, it send this message to notice its neighbors.
- GREY: Once a node is colored grey, it send this message to notice its neighbors.

Algorithm 4 Distibutedly Constructing d-MIS

Send HELLO messages to neighbors within $2d$ hops to initialize the local variables;
`In each round, for every white node` v`:`
Compare the value of $degree(v)$ with $degree(u)$, where $u \in N^{2d}(v)$. Use node id to break ties;
if *$degree(v)$ is the largest* **then**
  Color v black;
  Send a BLACK message to each node in $N^{2d}(v)$;
`For every node` v`, when receiving a BLACK message from node` u`:`
Update u's color in $neighbor\_info(v)$ or $neighbor\_2d\_info(v)$;
if *$color(v)$ is white and the distance between u and v is at most d hops* **then**
  Color it grey;
  Send a GREY message to each node in $N^{2d}(v)$;
`For every node` v`, when receiving GREY message from node` u`:`
Update u's color in $neighbor\_info(v)$ or $neighbor\_2d\_info(v)$.

Before Alg. 4, each node is colored white with a unique id. Initially, each node sends HELLO messages to its neighbors within $2d$ hops, so that it can get the initial values of its local variables. Afterwards, Alg. 4 is processed round by round. In each round, for any white node u, if $degree(u)$ is the maximum compared with its competitors, u is chosen as a dominator and colored black. Then, u's white neighbors within d hops are colored grey.

Since Alg. 4 is a distributed algorithm, there may be more than one node to be selected in each round. Besides, if a white node is selected and colored black, it will affect the states of nodes within its d hops, including the values of $degree(\cdot)$. Considering that, for any node u, we restrict that only one white node in $N^{2d}(u)$ is selected as a dominator in one round. In order to achieve that purpose, for each white node v, let its competitors be all the white nodes in $N^{2d}(v)$.

Next, we use an example to illustrate Alg. 4. For the 2-hop MCDS problem in Fig. 1, the procedure of Alg. 4 is as follows. Initially, $degree(4)$ and $degree(11)$ are the largest respectively, compared with nodes which are at most 4 hops away from them. Thus, we color node 4 and 11 black and all

their neighbors within 2 hops {1, 2, 3, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15} grey. In the second round, both of two white nodes (node 16 and node 17) have one white neighbor. We break tie with smaller id, color node 16 black and color node 17 gray. Since there are not more white nodes, Alg. 4 terminates.

B. Connect d-MIS

After Alg. 5, we have already selected a d-MIS. In this subsection, we introduce how to connect them with a distributed algorithm. Similar with Alg. 2, we connect the d-MIS with the method of divide and conquer. In each subprocedure, we select connectors based on the cost function as mentioned in Section IV-B.1. The details are shown in Alg. 5.

Messages involved in this phase are listed as follows.

1) COST_COMPUTE: Grey nodes use this kind of message to compute the value of its cost.
2) COST_INFO: Once a node figures out the value of its cost, it sends this kind of message to inform its neighbors.
3) COST_COMPARE: Grey nodes use this kind of message to compare its cost value with others.
4) BLUE: Once a node is colored blue, it sends this kind of message to inform its neighbors.
5) UPDATE_COMPONENT: This kind of message is used to update the value of $componentNum$.
6) COST_UPDATE: This kind of message is used to inform grey nodes to recompute its cost.

Since our algorithm is distributed, each node run the algorithm locally. As a consequence, in each iteration of some round, more than one node can be selected. Besides, once a node is selected, its state, such as its color, will changed. Such changes will further affect states of its neighbors. Hence, nodes selected in a same iteration may cause collisions. In order to avoid that, let each node maintain a local variable $componentNum$, which records the component number the node belongs to. The component here is actually r_i-hop connected component. Initially, at the beginning of each round in Alg. 5, for any node u, the value of $componentNum(u)$ equals to the id of node u. Then, each node informs its value of $componentNum(\cdot)$ within r_i hops. For any two nodes u, v, which are within r_i hops to each other, their $componentNum(\cdot)$ are set to be the smaller id of them.

Before the first iteration of the i^{th} round, for each grey node v, to compute $cost(v)$, node v sends messages to nodes in $N^{r_i}(v)$. Once a black or blue node receives such a message, it replies that with its value of $componentNum(\cdot)$ and the hop distance between itself and the sender of such message. With these information, node v can figure out $cost(v)$. After v figures out $cost(v)$, v broadcasts $cost(v)$ to nodes in $N^{r_i}(v)$. All black and blue nodes in $N^{r_i}(v)$ store $cost(v)$ in their local variables $connector(\cdot)$.

In each iteration of the i^{th} round, each grey node v compares $cost(v)$ with elements in $connector(w)$ where w is a black or blue node and $w \in N^{r_i}(v)$. If $cost(v)$ is the largest, it is selected as a connector and colored blue. Once a node v is selected as a connector, all the black and blue nodes in $N^{r_i}(v)$ set their $componentNum(\cdot)$ to be the id of node v. Of course, once a black or blue node u changes the value of $componentNum(u)$, it informs all those nodes, which are located in the same component with node u, to set

Algorithm 5 Distributely Connecting d-MIS

$r_0 = 2d + 1$;
In the i^{th} round:
$r_i = \lfloor \frac{r_{i-1}+1}{2} \rfloor$;
Initialize $componentNum.(v)$;
For every grey node $v \in V$:
Send COST_COMPUTE messages to nodes in $N^{r_i}(v)$ and compute $cost(v)$ based on those replies;
After $cost(v)$ is figured out, send COST_INFO messages to nodes in $N^{r_i}(v)$;
For each iteration in this round:
Send COST_COMPARE messages to nodes in $N^{r_i}(v)$;
if *All the replies are "YES"* **then**
  Color v blue;
  Send BLUE messages to nodes in $N^d(v)$;
When receiving a COST_UPDATE message:
Send COST_COMPUTE messages to nodes in $N^{r_i}(v)$ to compute $cost(v)$;
After $cost(v)$ is figured out, send CONT_INFO messages to nodes in $N^{r_i}(v)$;
For any black or blue node $u \in V$:
When receiving a COST_COMPUTE message from node v:
Reply with $componentNum(u)$ and the hop distance between u and v;
When receiving a COST_INFO message from node v:
Store $cost(v)$ in $connnector(u)$;
When receiving a COST_COMPARE message from a grey node v with parameter $cost(v)$:
If $cost(v)$ is the largest among all elements in $connector(u)$, reply with "YES". Use node id to break ties in the procedure of comparison. Otherwise, reply with "NO";
When receiving a BLUE message from node v with parameter $v(id)$:
Update v's color in u's local variables;
if *the distance between v and u is at most r_i* **then**
  Change $componentNum(u)$ to $v(id)$;
  Send UPDATE_COMPONENT with parameter $v(id)$ to nodes in $N^{r_i}(u)$, not including v, telling them to change their $compoonentNum$;
  Send COST_UPDATE to nodes in $N^{r_i}(u)$, telling them to recompute their $cost$;
When receiving a UPDATE_COMPONENT message from node v with parameter num:
Change $componentNum(u)$ to num;
Send UPDATE_COMPONENT with parameter num to nodes in $N^{r_i}(u)$, not including v.

their $componentNum(\cdot)$ as the current $componentNum(u)$. Besides, the new component informs its all grey neighbors within r_i hops to update their $cost(\cdot)$ and further update some values of $connector(\cdot)$.

The pseudo-codes of this procedure are presented in Alg. 5. For the 2-hop MCDS problem in Fig. 1, the procedure of Alg. 5 is as follows. At the beginning of the first round, there are three 3-hop connected components ({4}, {11}, {16}). We should 3-hop connect these three components into one. At this time, the values of nodes 7, 8, 9, 17 are 1/3, 2/5, 2/5, 1/3 respectively, while all other grey nodes have value 0. Since node 8 has smaller id than node 9, we color node 8 blue. Now the 3-hop connected component is only {4, 8, 11, 16}, and thus the first round terminates. Next, in the second round there are two 2-hop connected components ({4, 8, 16}, {11}). At this time nodes 9, 10 have the same value of 1/2 while other grey nodes have value 0. Similarly we color node 9 blue and terminate the second round. When the third round starts, there are four 1-hop connected components ({4}, {8, 9}, {11}, {16}), while nodes 7, 17, 10 have the same value of 1 and other grey nodes have value 0. Thus, we select node 7 in the first iteration of this round. Then, select 10 in the second iteration, and 17 in the third iteration. After that, Round 3 terminates. At this time $r = 1$. Consequently, Alg. 5 completes its task.

C. Remove Redundant Nodes

As we discussed in Section IV-C, there may exist a lot of redundant nodes after Alg. 4 and Alg. 5. In this section, we discuss how to remove those redundant nodes to further reduce the size of d-CDS. Similar with Section IV-C, we still only consider black leaf nodes in $G[C]$, where C is the d-CDS obtained from phase 1 and phase 2.

Messages involved in this phase are listed as follows.

1) ALTER_DOMINATOR: This kind of message is used to request whether their dominatees have alternative dominators.
2) GREY2: Once a dominator is removed and colored grey, it sends this message with its id to notice its neighbors.
3) CANDIDATE: Once a node is checked to be redundant, it sends this message with its id to notice its competitors within $2d$ hops.

Obviously, in this phase, the most important work is to decide whether a black node in C is redundant. According to the analysis Section IV-C, we just need to check the first requirement. As Alg. 6 shows, to check that, for any black node v which connects to only one blue node, v sends ALTER_DOMINATOR messages to its grey neighbors within d hops and asks whether they have alternative dominators. Once a grey node u receives such a message, it will check its local variable $neighbor\_info$ which includes all the nodes which can dominate u within d hops. If $neighbor\_info$ contains multiple dominators, u sends a positive reply. Otherwise, u sends a negative reply. Eventually, if v does not receive any negative message, it means v is a redundant node.

Since our algorithm is distributed, more than one node can be removed in each round. If a dominator is removed, it will affect the states of its d-hop neighbors. Considering that, there may exist some conflicts. For example, at the beginning of some round, assume that node t is dominated by node v and node u. And both node u and node v have only one dominatee, namely node t. Hence, in this round, both u and v are redundant. However, if we remove both of them, node t will have no dominator. To avoid such conflicts, for any node u, we restrict that only one dominator in $N^{2d}(u)$ can be removed in a round as Alg. 6 shows.

Algorithm 6 Removing Redundant Dominators

In each round, for every black node v which connects to only one blue node:
1 v sends ALTER_DOMINATOR messages to grey nodes in $N^d(v)$;
2 **if** *v does not receive any negative reply* **then**
3 Send CANDIDATE with v's id to nodes in $N^{2d}(v)$;
4 **if** *v does not receive any CANDIDATE message those id is larger than $id(v)$* **then**
5 Color v as grey;
6 Send GREY2 messages with v's id to nodes in $N^d(v)$;

In each round, for every grey node u:
When receiving ALTER_DOMINATOR message:
7 Check $neighbor\_info(u)$. If there are multiple dominators, reply with "YES". Otherwise, reply with "NO";

When receiving GREY2 message with id:
8 Update u's local variables;

In each round, for every blue node u:
When receiving GREY2 message with id:
9 **if** *the distance between u and the node corresponding with id is one hop* **then**
10 **if** *u connects to only one blue node* **then**
11 color v as black;

For the 2-hop MCDS problem in Fig. 1, the procedure of Alg. 6 is as follows. In the first iteration, there are three black nodes, nodes 4, 11, 16. By sending ALTER_DOMINATOR messages, only node 16 figures out that itself is redundant and send CANDIDATE messages. Since node 4 and node 11 are not redundant, they will not send CANDIDATE messages, thus node 16 does not receive any CANDIDATE messages. Hence, node 16 is colored grey and send GREY2 messages. Afterwards, blue node 17 receives such GREY2 message and the distance between itself with node 16 is one hop. Thus, node 16 is colored black. Similarly, in the second iteration, there are three black nodes, nodes 4, 17, 11. Node 17 is redundant and remove it. Since the blue neighbor of node 17 (node 8) connects to two blue nodes, it does nothing. Consequently, there are two black nodes, nodes 4, 11. By checking, neither of them is redundant. Thus, Alg. 6 terminates.

VI. Performance Analysis

A. An Upper Bound of d-MIS Size for a UDG G

To analyze the performance of CS-Cluster, we need to firstly discuss the upper bound of d-MIS size for a given graph. Given

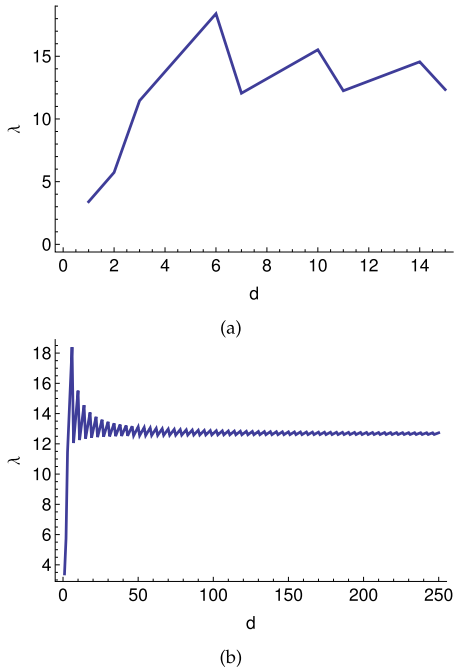


Fig. 5. The relation between λ and d. (a) The value of d is from 1 to 15. (b) The value of d is from 1 to 250.

a UDG $G = (V, E)$, let $\alpha(G)$ denote the size of d-MIS in G and $\gamma(G)$ the size of d-MCDS in G, and let λ be the ratio of $\alpha(G)$ and $\gamma(G)$. λ is computed as follows.

$$\lambda = \begin{cases} \dfrac{\Delta S}{\sqrt{3}/2}, & \text{if } d \le 2, \\ \dfrac{\Delta S}{\frac{\sqrt{3}}{2}\lceil \frac{1}{2}\lfloor \frac{d-1}{2} \rfloor \rceil} + 3.399, & \text{if } d \ge 3. \end{cases} \tag{2}$$

You can find the proof of λ in the Appendices A and B.[1]

Fig. 5 depicts the relation between λ and d. From these figures, we can conclude that the value of λ is maximum when $d = 6$ and is equal to 18.4. Moreover, we also find when $d \to \infty$, $\lambda \to 12.6366$. Therefore, for arbitrary value of d, the value of λ is no more than 18.4. In this sense, λ has little relation to d. Thus, our analysis improve the previous $O(d)$ into $O(1)$. This is a huge improvement.

B. The Approximation Ratio of CS-Cluster

Assume the size of an optimal d-MCDS is opt. Through Alg. 4, we get a d-MIS, namely the set M. According to the analysis in Appendix B, we have $|M| \le \lambda opt$. As for Alg. 5, we have Lemma 4 below.

Lemma 4: The total number of connectors selected in Alg. 5, namely $C \backslash M$, is at most $2d\lambda opt$.

Proof: In the first round, as we described above, Alg. 5 is to connect all $r(1)$-hop connected components into one $r(1)$-hop connected component. Moreover, before the end of the current round, there always exists one node that can $r(1)$-hop connect two $r(1)$-hop connected components. Let

[1]Appendices are shown online as the supplemental materials, available on IEEE Xplore.

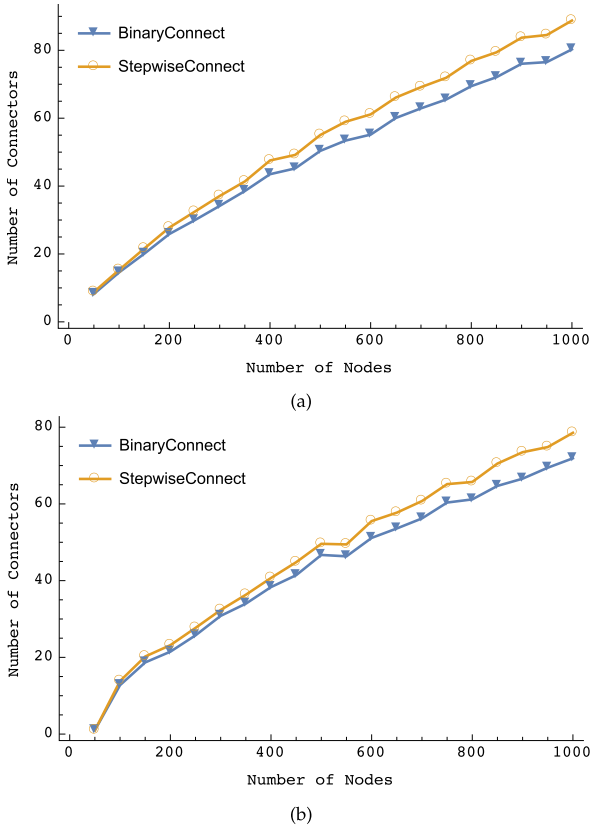


Fig. 6. Number of connectors with different connecting ways. (a) d=3. (b) d=5.

v be that node. Then we have

$$cost(v) \ge \frac{1}{r(0) - 1} = \frac{1}{2d}.$$

Since Alg. 5 always selects the node with largest value in each iteration, the node that is chosen in this iteration has the value of at least $cost(v)$. Consequently, for each node in C_1, the value is at least $\frac{1}{2d}$ when it is chosen.

There exists an alternative algorithm to connect M. For each node $v \in C_1$, when it is chosen, assume the number of $r(1)$-hop connected components it can $r(1)$-hop connect is $comp(v)$. We select all the nodes in the shortest paths from v to the $comp(v)$ $r(1)$-hop connected components. Obviously, all those chosen nodes can connect M into a d-CDS. Besides, the size of C_i is at most $|M| - 1$ because each node in C_i can reduce at least one $r(1)$-hop component and the number of the original $r(1)$-hop connected components is at most $|M|$ when Alg. 5 begins. Thus, the size of these connectors is

$$\sum_{v \in C_1} no.(v) \le \sum_{v \in C_1} \frac{comp(v)}{cost(v)} \le 2d|M|.$$

Similarly, in the i^{th} round, for each node in C_i, the value when it is chosen is at least $\frac{1}{r(i-1)-1}$. The initial number of $r(i)$-hop connected components is no more than

$$|M| + \sum_{j=1}^{i-1} |C_j| \le 2^{i-1}|M|.$$

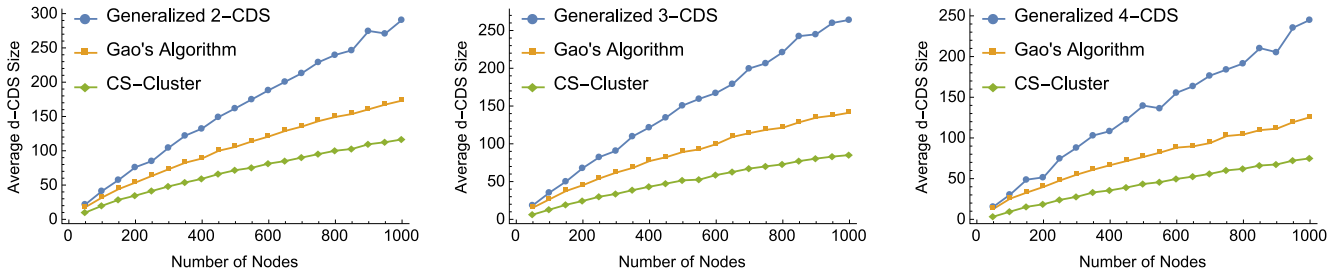


Fig. 7. Comparison among CS-Cluster, Gao's algorithm in [29] and Generalized d-CDS in [27] in different cases where $d = 2, 3, 4$ respectively. (a) Comparison when $d = 2$. (b) Comparison when $d = 3$. (c) Comparison when $d = 4$.

There also exists an alternative algorithm to connect $M \cup_{j<i} C_i$ into a d-CDS, and the size of the connectors is

$$\sum_{v \in C_i} no.(v) \leq \sum_{v \in C_i} \frac{comp(v)}{cost(v)} \leq (r(i-1)-1)2^{i-1}|M|.$$

Since

$$\begin{aligned}(r(i-1)-1)2^{i-1} &= (\lfloor \frac{r(i-2)+1}{2} \rfloor - 1) \cdot 2 \cdot 2^{i-2} \\ &\leq (r(i-2)-1)2^{i-2} \\ &\quad \cdots \\ &\leq r(0) - 1 = 2d,\end{aligned}$$

we have $\sum_{v \in C_i} no.(v) \leq 2d|M|$.

Finally, in the t^{th} round where $r(t) = 1$, Alg. 5 does the same with its corresponding algorithm. Therefore, $|C \backslash M| = \sum_{v \in C_t} no.(v) \leq 2d|M|$, and Lemma 4 follows. □

From the procedure of proof above, it not difficult to conclude the following corollary.

Corollary 1: After the i^{th} round, Alg. 5 will indicate a feasible solution F_i with size $\sum_{v \in C_i} no.(v)$ which alone can connect M into d-CDS. Moreover, the size of F_i is no more than F_{i-1}.

With Lemma 4, we can finally get the following theorem.

Theorem 2: Our three-phase algorithm CS-Cluster has an approximation ratio of $(2d + 1)\lambda$.

C. Further Discussion

In this subsection, we will discuss some extra advantages of CS-Cluster as follows.

Firstly, although what we discussed above is under UDG model, it is not difficult to see that CS-Cluster actually fits for general graph model and even for heterogeneous networks.

Secondly, it is not difficult to find that the second phase of CS-Cluster is also a Steiner algorithm. Although Alg. 2 is only fit for connecting d-MIS, it can also be applied to genal Steiner problem with slightly modification.

Besides, as we can see, the definition of $r(i)$ in Alg. 5 can greatly affect the performance of CS-Cluster. In the current Alg. 5, we connect those d-MIS nodes in a logarithmic way. This can of course increase the speed of Alg. 5. Then, an intuitive idea to improve the performance of CS-Cluster is to use a linear way to connect those d-MIS nodes. Correspondingly, the definition of $r(i)$ is as follows:

$$\begin{cases} r(0) = 2d + 1, \\ r(i) = r(i-1) - 1. \end{cases} \tag{3}$$

In this new way, the performance of our algorithm may be increased at the cost of time complexity. However, what we expect is not right. Applying Eqn. (3) to Alg. 2, we get the results as Fig. 6 shows.

From Fig. 6, we can see that CS-Cluster uses more connectors to connect d-MIS in the linear connection way than the original logarithmic connection way.

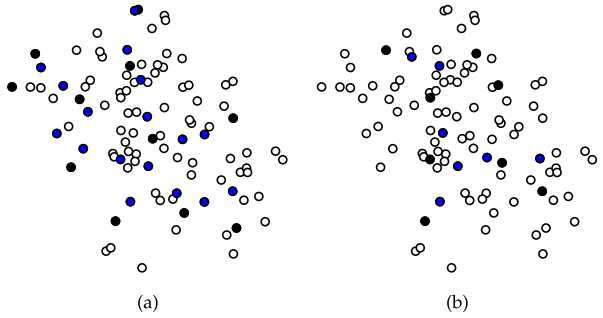


Fig. 8. A sample of UDG which contains 100 nodes. (a) 2-CDS by Gao's algorithm. (b) 2-CDS by CS-Cluster

VII. Simulation

In this section, we compare CS-Cluster with previous literature [27], [29] (refer as Gao's algorithm and Generalized d-CDS). In our simulations, we randomly deploy sensor nodes in a 2D virtual space. The number of nodes varies from 50 to 1000 at intervals of 50. Each two nodes can connect to each other when the distance between them is at most 1. Moreover, we also ensure that the generated graph is connected. In fact, Generalized d-CDS generates d-connected set instead of connected set (two vertices are d-connected if there is a path of length smaller than d between them), therefore we modified Generalized d-CDS (connecting separate vertices via shortest path) to generate a connected set.

Fig. 7(a)-(c) exhibit the performance comparisons between these algorithms with different d. From the figure, we can

conclude that CS-Cluster always outperforms the others under different settings.

Fig. 8 shows a comparison between CS-Cluster and Gao's algorithm with 100 nodes where $d = 2$. We can see the superiority of CS-Cluster since Gao's algorithm constructs a 2-CDS with 28 nodes while CS-Cluster only uses 15 nodes.

VIII. Conclusion

In this paper, we studied the multi-hop connected clustering problem for a given homogenous wireless network, which can be formed as finding a minimum d-hop connected dominating set problem (d-MCDS) for a given graph. We then proposed a distributed approximation algorithm named *Connected Sparse Clustering Scheme* (CS-Cluster) to solve the problem. CS-Cluster consists of three phases: dominator selection, connector insertion, and redundancy elimination. To evaluate the performance of CS-Cluster, we estimated the upper bound λ for the dominator size in a unit disk graph, and proved that λ is no more than 18.4. As a result, we reduce the bound of $O(d)$ from previous literature [30] to $O(1)$, and achieved an approximation ratio of $(2d+1)\lambda$ for CS-Cluster, which is the best constant-factor approximation up to now. Our simulation results also exhibited the outstanding performance of CS-Cluster.

References

[1] J. Wang, Z. Cao, X. Mao, and Y. Liu, "Sleep in the dins: Insomnia therapy for duty-cycled sensor networks," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 1186–1194.

[2] S. Guo, C. Wang, and Y. Yang, "Mobile data gathering with Wireless Energy Replenishment in rechargeable sensor networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1932–1940.

[3] Q. Liao *et al.*, "Visualizing anomalies in sensor networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 460–461, 2011.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[5] X. Fang, H. Gao, J. Li, and Y. Li, "Approximate multiple count in wireless sensor networks," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 1474–1482.

[6] C. Schurgers and M. B. Srivastava, "Energy efficient routing in wireless sensor networks," in *Proc. IEEE MILCOM*, vol. 1. Oct. 2001, pp. 357–361.

[7] C. R. Dow, P. J. Lin, S. C. Chen, J. H. Lin, and S. F. Hwang, "A study of recent research trends and experimental guidelines in mobile ad-hoc network," in *Proc. IEEE AINA*, vol. 1. Mar. 2005, pp. 72–77.

[8] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *Proc. IEEE INFOCOM*, vol. 3. Mar./Apr. 2003, pp. 1713–1723.

[9] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach," in *Proc. IEEE INFOCOM*, Mar. 2004, p. 640.

[10] J. Ma, W. Lou, Y. Wu, M. Li, and G. Chen, "Energy efficient TDMA sleep scheduling in wireless sensor networks," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 630–638.

[11] Y. Zhuang, J. Pan, and L. Cai, "Minimizing energy consumption with probabilistic distance models in wireless sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[12] M. Alaei and J. M. Barcelo-Ordinas, "Node clustering based on overlapping FoVs for wireless multimedia sensor networks," in *Proc. IEEE WCNC*, 2010, pp. 1–6.

[13] O. Younis and S. Fahmy, "HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Trans. Mobile Comput.*, vol. 3, no. 4, pp. 366–379, Oct./Dec. 2004.

[14] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 32–41.

[15] Y. Fernandess and D. Malkhi, "k-clustering in wireless ad hoc networks," in *Proc. ACM POMC*, 2002, pp. 1–7.

[16] F. G. Nocetti, J. S. Gonzalez, and I. Stojmenovic, "Connectivity based k-hop clustering in wireless networks," *Telecommun. Syst.*, vol. 22, no. 1, pp. 205–220, Jan. 2003.

[17] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Ann. Discrete Math.*, vol. 48, pp. 165–177, Jan. 1991.

[18] P.-J. Wan, K. M. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in *Proc. IEEE INFOCOM*, Jun. 2002, pp. 1597–1604.

[19] W. Wu, H. Du, X. Jia, Y. Li, and S. C.-H. Huang, "Minimum connected dominating sets and maximal independent sets in unit disk graphs," *Theor. Comput. Sci.*, vol. 352, nos. 1–3, pp. 1–7, Mar. 2006.

[20] X. Gao, Y. Wang, X. Li, and W. Wu, "Analysis on theoretical bounds for approximating dominating set problems," *Discrete Math., Algorithms Appl.*, vol. 1, no. 1, pp. 71–84, 2009.

[21] M. Li, P.-J. Wan, and F. Yao, "Tighter approximation bounds for minimum CDS in unit disk graphs," *Algorithmica*, vol. 61, no. 4, pp. 1000–1021, Dec. 2011.

[22] Y. L. Du and H. W. Du, "A new bound on maximum independent set and minimum connected dominating set in unit disk graphs," *J. Combinat. Optim.*, vol. 30, no. 4, pp. 1173–1179, Nov. 2015.

[23] D. Kim *et al.*, "A better approximation algorithm for computing connected dominating sets in unit ball graphs," *IEEE Trans. Mobile Comput.*, vol. 9, no. 8, pp. 1108–1118, Aug. 2010.

[24] T. H. P. Vuong and D. T. Huynh, "Adapting d-hop dominating sets to topology changes in ad hoc networks," in *Proc. IEEE ICCCN*, Oct. 2000, pp. 348–353.

[25] T. N. Nguyen and D. T. Huynh, "Connected D-hop dominating sets in mobile ad hoc networks," in *Proc. IEEE WiOpt*, Feb./Mar. 2006, pp. 1–8.

[26] D. Cokuslu and K. Erciyes, "A hierarchical connected dominating set based clustering algorithm for mobile ad hoc networks," in *Proc. IEEE MASCOTS*, Oct. 2007, pp. 60–66.

[27] M. Q. Rieck, S. Pai, and S. Dhar, "Distributed routing algorithms for multi-hop ad hoc networks using d-hop connected d-dominating sets," *Comput. Netw.*, vol. 47, no. 6, pp. 785–799, Apr. 2005.

[28] X. Li and Z. Zhang, "Two algorithms for minimum 2-connected r-hop dominating set," *Inf. Process. Lett.*, vol. 110, no. 22, pp. 986–991, Oct. 2010.

[29] X. Gao, W. Wu, X. Zhang, and X. Li, "A constant–factor approximation for d–hop connected dominating sets in unit disk graph," *Int. J. Sensor Netw.*, vol. 12, no. 3, pp. 125–136, 2012.

[30] Z. Zhang, Q. Liu, and D. Li, "Two algorithms for connected r-hop k-dominating set," *Discrete Math., Algorithms Appl.*, vol. 1, no. 4, pp. 485–498, 2009.

[31] X. Zhu, J. Li, Y. Xia, X. Gao, and G. Chen, "An efficient distributed node clustering protocol for high dimensional large-scale wireless sensor networks," in *Proc. ACM ICUIMC*, vol. 4. 2014, pp. 1–8.

[32] W. Wang, D. Kim, N. Sohaee, C. Ma, and W. Wu, "A PTAS for minimum d-hop underwater sink placement problem in 2-D underwater sensor networks," *Discrete Math., Algorithms Appl.*, vol. 1, no. 2, pp. 283–289, 2009.

[33] D. Kim *et al.*, "Minimum data-latency-bound k-sink placement problem in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 5, pp. 1344–1353, Oct. 2011.

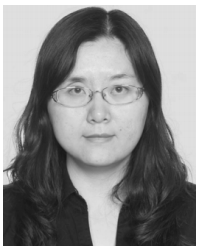
[34] W. Wang *et al.*, "A new constant factor approximation to construct highly fault-tolerant connected dominating set in unit disk graph," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 18–28, Feb. 2017.

[35] B. Liu *et al.*, "On approximating minimum 3-connected m-dominating set problem in unit disk graph," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2690–2701, Oct. 2016.

[36] W. Wang *et al.*, "On construction of quality fault-tolerant virtual backbone in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1499–1510, Oct. 2013.

[37] Y. Hong *et al.*, "Construction of higher spectral efficiency virtual backbone in wireless networks," *Ad Hoc Netw.*, vol. 25, pp. 228–236, Feb. 2015.

[38] D. Kim, Y. Wu, Y. Li, F. Zou, and D. Z. Du, "Constructing minimum connected dominating sets with bounded diameters in wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 2, pp. 147–157, Feb. 2009.



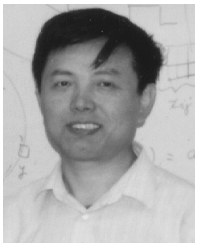
Xiaofeng Gao received the B.S. degree in information and computational science from Nankai University, China, in 2004, the M.S. degree in operations research and control theory from Tsinghua University, China, in 2006, and the Ph.D. degree in computer science from The University of Texas at Dallas, USA, in 2010. She is currently an Associate Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. She has authored over 100 peer-reviewed papers in the related area, including well-archived international journals, such as the IEEE TRANSATIONS ON COMPUTERS, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON MOBILE COMPUTING, and also in well-known conference proceedings, such as INFOCOM, SIGKDD, and ICDCS. Her research interests include wireless communications, data engineering, and combinatorial optimizations. She has served on the Editorial Board of *Discrete Mathematics* and *Algorithms and Applications*.

Xudong Zhu received the B.S. degree in computer science from Shanghai Jiao Tong University in 2014. He is currently a Graduate Student with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include combinatorial optimization and approximation algorithm.

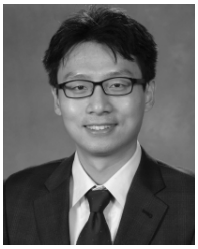
Jun Li received the B.S. degree in optical information science and technology from Shanghai Jiao Tong University in 2013. He is currently a Graduate Student with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include connected dominating set in wireless sensor works, coverage in wireless sensor works, and approximation algorithm.

Fan Wu received the B.S. degree in computer science from Nanjing University in 2004 and the Ph.D. degree in computer science and engineering from The State University of New York at Buffalo in 2009. He is currently an Associate Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He has visited the University of Illinois at Urbana–Champaign as a Post-Doctoral Research Associate. He has authored over 100 peer-reviewed papers in technical journals and conference proceedings. His research interests include wireless networking and mobile computing, algorithmic game theory and its applications, and privacy preservation. He has served on the Editorial Board of *Computer Communications*, and as a member of technical program committees of more than 60 academic conferences.

Guihai Chen received the B.S. degree from Nanjing University in 1984, the M.E. degree from Southeast University in 1987, and the Ph.D. degree from The University of Hong Kong in 1997. He had been invited as a Visiting Professor by many universities, including the Kyushu Institute of Technology, Japan, in 1998, the University of Queensland, Australia, in 2000, and Wayne State University, USA, from 2001 to 2003. He is currently a Distinguished Professor with Shanghai Jiao Tong University, China. He has authored over 400 peer-reviewed papers, and over 200 of them are in well-archived international journals or conference proceedings. He has a wide range of research interests including sensor network, peer-to-peer computing, high-performance computer architecture, and combinatorics.



Ding-Zhu Du received the M.S. degree from the Chinese Academy of Sciences in 1982 and the Ph.D. degree from the University of California at Santa Barbara in 1985, under the supervision of Prof. Ronald V. Book. He is a Professor with the Department of Computer Science, University of Texas at Dallas. He was with the Mathematical Sciences Research Institute, Berkeley, USA, from 1985 to 1986, the Department of Mathematics, Massachusetts Institute of Technology, USA, from 1986 to 1987, the Department of Computer Science, Princeton University, USA, from 1990 to 1991, and the Department of Computer Science and Engineering, University of Minnesota from 1992 to 2005. He is the Editor-In-Chief of the *Journal of Combinatorial Optimization*. He is on the editorial boards for several other journals.



Shaojie Tang received the Ph.D. degree in computer science from the Illinois Institute of Technology in 2012. He is currently an Assistant Professor with the Naveen Jindal School of Management, The University of Texas at Dallas. His research interest includes social networks, mobile commerce, game theory, e-business, and optimization. He received the Best Paper Awards in the ACM MobiHoc 2014 and the IEEE MASS 2013. He also received the ACM SIGMobile Service Award in 2014. He served in various positions, such as as chairs and TPC members, at numerous conferences, including the ACM MobiHoc and the IEEE ICNP. He is an Editor of *Information Processing in the Agriculture* and the *International Journal of Distributed Sensor Networks*.