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

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

*Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, P.R.China
†University of Texas at Dallas, Richardson, TX 75080, USA

Abstract—Wireless sensor networks (WSNs) have been widely used in 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 algorithm, named CS-Cluster, to address the $d$-MCDS problem. 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 $\lambda$ 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.

I. INTRODUCTION

Wireless sensor network (WSN) is a kind of self-organized communication system consisting of many small-sized, inexpensive, and battery-powered sensors. Since such sensors can be remotely deployed in large numbers and operated autonomously in unattended environments, 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].

Because the unattended environments of WSNs generally make it quite difficult to recharge sensor batteries, sensors in such environments are energy constrained. Therefore, energy conservation is always an important factor for extending the lifetime of WSNs. In the literature, a lot of researches have been done to extend the lifetime of WSNs [6], [8]–[11]. In addition, sensor nodes also have constraints on communication bandwidth, communication range, and storage space. The communication range restricts the transmission capacity of a sensor within a localized area in the network. 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 given WSN into several disjointed clusters, each with a cluster head to take charge of the data gathering process and the communications inside or outside the cluster. In this mechanism, any ordinary node in the WSN only needs to collect information and send the data to its corresponding cluster head, which saves a lot of energy by avoiding many redundant message forwarding processes. The most important part for clustering 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 to efficiently reduce the redundant sensing area, the authors proposed a clustering algorithm to optimize the energy conservation. Load balancing is also a crucial issue for clustering mechanisms. Younis et al. [13] aimed at setting equal-sized clusters to balance the work load of each cluster. By setting a hop threshold $k$, researchers also discussed $k$-hop clustering problem [14]–[16] requiring that each node in the network should not be more than $k$ hops far 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 downgrade the performance of clustering strategy. An effective way to control the average size is to set a parameter $d$, and to let each cluster head only take charge of its $d$-hop neighbors. To achieve better communication between clusters, any pair of the cluster heads should be able to communicate with each other directly or through the relay of the 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 $(u, v) \in E$ if $u$ and $v$ communicate with each other. In this 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 a 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 $d$-MCDS problem, and prove that its approximation ratio is $(2d + 1)\lambda$, where $\lambda$ 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 $\lambda$. 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 IV, we describe our algorithm CS-Cluster for $d$-MCDS problem. In Section V, we analyze the upper bound of $d$-MIS for a given graph. 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 $(u, v) \in E$ if and only if $\text{dis}(u, v) \leq 1$. Here $\text{dis}(u, v)$ 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 within $d$-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 within $d$-hop between $u$ and $v$.

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

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 $v \in V \setminus I$, $I \cup \{v\}$ 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

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

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 $d$-MCDS problem as follows.

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

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, 1-MCDS problem) is a classic NP-complete problem. Over the past decades, there are a lot of researches related to it. In [17], Clark et al. first proved that MCDS problem 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 part of their algorithm is...
to select an MIS. Then, it adds some extra nodes to connect the MIS into a CDS. Later, a lot of related works were done to improve the approximation ratio of this two-phase algorithm [19]–[21]. The ratio of the size of MIS and 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 the theoretical bound is 3.399 by Du et al. [22].

The problem of d-MCDS is an extension of 1-MCDS problem. In 2000, Vuong et al. [23] proved that d-MCDS is NP-complete in general graph by a reduction from 3-SAT problem. Later, Nguyen et al. [24] proved that d-MCDS is NP-complete even in UDG. A lot of heuristic algorithms were proposed to find a feasible solution of d-MCDS [25], [26]. However, these algorithms all lacked approximation analysis.


There are many other related works for the clustering problem. Wang et al. [31] presented a PTAS to minimize the average hop distance from any nodes to cluster heads in 2D sensor networks without connectivity property. Kim et al. [32] discussed how to find the locations of k sinks such that the maximum distance from other nodes to sinks is minimized.

IV. CS-Cluster for d-MCDS problem

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

A. Constructing a d-MIS

According Lemma 1, we can select a d-MIS as d-DS for a given graph. And we usually 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 [28]–[30], researchers often restrict that the new node is $d+1$ hops away from $S$. However, if we remove such restriction, we can form a sparser d-MIS. An example can be shown in Fig. 1 where $d = 2$. If we implement traditional algorithms, we will select nodes $\{4, 9, 12, 15, 16\}$ 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 propose an efficient distributed algorithm to select a sparser d-MIS as Alg. 1 shows. The main idea is to select the $d$-hop dominator as far as possible to the current $d$-IS. This strategy is also helpful to select connectors more flexibly in the next step.

In Alg. 1, we use $\text{dominatee}(u)$ to denote the number of $u$’s white neighbors within $d$ hops in the current round. This algorithm is processed round by round. Obviously, for any white node $u$, if $\text{dominatee}(u)$ is the maximum compared with those values of nodes in $N^d(u)$, selecting node $u$ can cover the most white nodes compared with selecting any node in $N^d(u)$. Since Alg. 1 is a distributed algorithm, there may be more than one nodes to be selected in each round. Once a white node is selected and is colored black, it may affect the states of nodes within its $d$ hops, including the value of $\text{dominatee}()$. Considering that, for any node $u$, we restrict that only one white node in $N^d(u)$ is selected in one round.

Besides, we use set $\text{dominator}(u)$ to denote the nodes which can dominate node $u$. These variable will be used in later. The pseudo-code of this procedure is presented in Alg. 1.

Algorithm 1: Constructing d-MIS

```
In each round, for every white node $v$ in $V$:
1. Compare the value of $\text{dominatee}(v)$ with $\text{dominatee}(u)$, where $u \in N^d(v)$.
   Use node id to break ties in the procedure of comparison;
2. if $\text{dominatee}(v)$ is the largest then
   1. Color $v$ black;
   4. Send message to each node $u$ in $N^d(v)$, and tell it to add $v$’s id to $\text{dominator}(u)$.
      If $u$ is white, color it grey;
   5. Send message to each node $w \in N^d(v)$, and tell it to check and update the value of $\text{dominatee}(u)$;
```

Fig. 2 shows the corresponding 2-MIS after processing Alg. 1 for the 2-hop MCDS problem in Fig. 1.
B. Connecting a d-MIS

After Alg. 1, we get a d-MIS and denote it as $M$, which is also a d-DS. In this subsection, we discuss how to connect it into a d-CDS. Before introducing the algorithm, we first introduce some notations and definitions that will be used in this section.

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 $k$. A subset $S \subseteq V$ is a "$k$-hop connected $d$-hop dominating set" iff:

- $S$ is a $d$-hop dominating set of graph $G$.
- For any vertex $u \in S$, there exists at least one vertex $v \in S$ such that $u$ and $v$ are $k$-hop connected.

In addition, for a graph $G = (V, E)$, a subset $S \subseteq V$ is a "$k$-hop connected component" if:

- 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 $v \in V \setminus S$, there does not exist any vertex $u \in S$ such that $u$ and $v$ are $k$-hop connected.

For a vertex subset $C \subseteq V$, let $f_k(C)$ be the number of $k$-hop connected components of the induced subgraph $G[C]$.

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

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

Next, we will start to introduce our algorithm which can connect $M$ into a d-CDS. The algorithm in this subsection is processed round by round. In the $i$th round, a subset $C_i$ is selected from $V \setminus \bigcup_{j < i} C_j \cup M$ such that $\bigcup_{j \leq i} C_j \cup M$ is a $r(i)$-hop connected $d$-hop dominating set. The definition of $r(i)$ is:

- $r(0) = 2d + 1$,
- $r(i) = \left\lceil \frac{r(i-1)+1}{2} \right\rceil$.

Obviously, after Alg. 1, $M$ is an $(2d+1)$-hop connected $d$-hop dominating set, which means any two nodes in $M$ are $(2d+1)$-hop connected in graph $G$. 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

1. $r_0 = 2d + 1$;
2. In the $i$th round:
3. Initialize $\text{componentNum}(v)$;
4. Send messages to nodes in $N^r_i(v)$ to compute $\text{cost}(v)$;
5. After $\text{cost}(v)$ is figured out, inform this value to nodes in $N^r_i(v)$;
6. For each iteration in this round:
7. Compare $\text{cost}(v)$ with elements in $\text{connector}(u)$ where node $u$ is a black or blue node and $u \in N^r_i(v)$. Use node $id$ to break ties in the procedure of comparison;
8. If $\text{cost}(v)$ is the largest then
9. Color $v$ blue;
10. Send messages to nodes in $N^d(v)$, inform them to add $v$’s id to their $\text{dominator}()$;
11. Send messages to nodes in $N^r_i(v)$, inform them to change $\text{componentNum}(\cdot)$ to be the id of node $v$, or to recompute the value of $\text{cost}(\cdot)$;
12. In any black or blue node $u \in V$:
13. When receiving a message informing to recompute $\text{cost}(\cdot)$;
14. Store $\text{cost}(v)$ in $\text{connector}(u)$;
15. When receiving a message from node $v$ informing the value of $\text{cost}(v)$;
16. When receiving a message from node $v$ informing to change $\text{componentNum}(\cdot)$ as it is required;
17. Inform $u$’s black and blue neighbors within $r_i$ hops, not including $v$, to change their $\text{componentNum}(\cdot)$ to be $\text{componentNum}(u)$;
18. Inform nodes in $N^r_i(u)$ to recompute their $\text{cost}(\cdot)$;

To make the size of $C_i$ as small as possible, we should choose the most efficient nodes that makes $\bigcup_{j \leq i} C_j \cup M$ be 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 add another requirement in this procedure.

To better understand our algorithm, we first assume that each node knows the global information. At the beginning of the $t$th iteration of the $i$th round, $C_i$ contains $t - 1$ nodes. Assume $C = \bigcup_{j \leq i} C_j \cup M$. For a node $v$ in $V \setminus C$, 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 $\text{no}(v)$ to denote the total number of nodes in those shortest paths except two end points (but includes $v$). Moreover, we use $\text{cost}(v)$ to denote the cost of $v$ under the current state and its definition is as follows.

$$\text{cost}(v) = \frac{-\Delta_v f_{r(i)}(C \cup C_i)}{\text{no}(v)}.$$ 

In every iteration, we select the node with the largest cost. Note that the additional requirement of $\text{cost}(v)$ is much crucial to guarantee and improve the performance of our algorithm.

Since our algorithm here is a distributed algorithm, each node run the algorithm locally. As a consequence, in each iteration of some round, there may be more than one node to be selected. Besides, once a node is selected, its state, such as its color, will changed. Such change will further affect states of its neighbors. Hence, nodes selected in a same iteration of some round may cause collisions. In order to avoid that, we need to do more than the simpler case above. First, each node has a local variable $\text{componentNum}(\cdot)$, which records the component the node belongs to. The component here is actually $r_i$-hop connected component. Initially, at the beginning of each round in Alg. 2, for any node $u$, the value of $\text{componentNum}(u)$ equals to the id of node $u$. Then, each node informs its value of $\text{componentNum}(\cdot)$ within $r_i$ hops. For any two nodes $u, v$, which are within $r_i$ hops to each other, their $\text{componentNum}(\cdot)$ are set to be the smaller id of them.

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

In each iteration of the $i^{th}$ round, each grey node $v$ compares $\text{cost}(v)$ with elements in $\text{connector}(w)$ where $w$ is a black or blue node and $w \in N^{r_i}(v)$. If $\text{cost}(v)$ is the largest, $v$ 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 $\text{componentNum}(\cdot)$ to be the id of node $v$. Of course, once a black or blue node $u$ changes the value of $\text{componentNum}(u)$, it informs all those whose $\text{componentNum}(\cdot)$ are the same with $\text{componentNum}(u)$ to set their $\text{componentNum}(\cdot)$ as $\text{componentNum}(u)$. Besides, the new component informs its all grey neighbors within $r_i$ hops to update their $\text{cost}(\cdot)$ and further update some values of $\text{connector}(\cdot)$.

An example of Alg. 2 is shown in Fig. 3.

Next, we prove the correctness of Alg. 2. Initially, the input $M$ is a $d$-MIS, so $f_{d+1}(M) = |M|$ and $f_{2d+1}(M) \leq |M|$. 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

Fig. 3. The result after processing Alg. 2. At the beginning of the first round, there are 3 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. 2 completes its task.

“(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 $f_{r(i)}(\bigcup_{j \leq i} C_j \cup M) = 1$.

**Proof:** When $i = 0$, Lemma 3 holds obviously as we have shown above. Assume Lemma 3 holds for the $i^{th}$ round where $i \leq 0$, 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 only one $r(i+1)$-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 $\left\lfloor \frac{r(i+1)}{2} \right\rfloor = r(i+1)$ 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 suitable 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
contains.\[16x-0\] Otherwise, sends a negative reply.\[16x-0\] It is crucial to estimate the performance of our\[16x-0\] algorithm, in this section we will discuss the upper bound of this ratio from two perspectives.

### A. Voronoi Division and Euler’s Formula

Assume that the d-MCDS of $G$ is $M$ and the d-MIS with maximum size is $I$. Obviously for any two nodes $u, v \in I$, the Euclidean distance between them is greater than 1. Thus, $d_{disk}(u)$ and $d_{disk}(v)$ do not intersect with each other. Moreover, for any node $u \in I$, $d_{disk}(u)$ can be completely contained in $\bigcup_{v \in M} d_{disk}(d+0.5)(v)$. Then the value of $d_{disk}(M)$ should be less than $A_{d+0.5}(M)/A_{0.5}(v)$. It is easy to figure out that

$$A_{d+0.5}(M) \leq \pi(d+\frac{1}{2})^2 + (\gamma(G) - 1)\Delta S \quad (1)$$

where

$$\Delta S = (\pi - 2\theta)(d+\frac{1}{2})^2 + (d+\frac{1}{2})\sin\theta,$$

$$\theta = \arccos\left(\frac{1}{2d+1}\right).$$

Consequently, we can get an upper bound of $\alpha(G)$. However, this upper bound is a very rough result and we can improve it by Voronoi Division.
With Voronoi Division, we can partition the region \( A_{d+0.5}(M) \) into Voronoi cells. For any node \( u \) in \( I \), there is a corresponding Voronoi cell \( V(u) \) which contains \( \text{disk}_{0.5}(u) \). There are two types of Voronoi cells. If a cell does not contain any boundary point of \( A_{d+0.5}(M) \), then we call it non-boundary Voronoi cell. Otherwise, it is a boundary Voronoi cell. For any non-boundary Voronoi cell, it is a polygon. Let \( s_k \) denote the minimum area of Voronoi cell with \( k \) edges. For those boundary Voronoi cells, we also consider them as a special kind of polygons and let \( s_k' \) denote the minimum area of boundary Voronoi cell with \( k \) edges. Next, we will first calculate the values of \( s_k \) and \( s_k' \).

Gao et al. [20] has proved that, if a non-boundary Voronoi cell \( V(u) \) has \( k \) edges and \( k \leq 6 \), then the regular polygon with \( k \) edges which is inscribed with \( \text{disk}_{0.5}(u) \) has the smallest area. Let \( P_k \) be such kind of polygon. If \( V(u) \) is a boundary Voronoi cell with \( k \) edges, under two conditions \( V(u) \) could have the minimum area. The first condition is when boundary arc cut off one edge of \( P_k \) as Fig. 5 shows, and we name it as \( E_k \). The second condition is when boundary arc cut off one angle of \( P_k \) as Fig. 5 shows, and we name it as \( A_{k+1} \).

\[ \text{Fig. 5. An example to show the conditions of non-boundary Voronoi cells and boundary Voronoi cells having the minimum area when } k = 6. \]

Since \( V(u) \) with 6 adjacent neighbors is the densest situation if any two small disks does not intersect with each other, any non-boundary Voronoi cell \( V(u) \) with more than 6 edges cannot be inscribed with \( \text{disk}_{0.5}(u) \) any more. Thus, the areas of that kind of Voronoi cells are greater than \( s_6 \). Consequently, we can conclude that \( s_i \geq s_6 \) for \( i \geq 7 \). Similarly, for \( i \geq 7 \), the area of \( E_i \) is greater than the area of \( E_6 \). And the area of \( A_i \) is greater than the area of \( A_7 \) for \( i \geq 8 \).

Since non-boundary Voronoi cells have nothing to do with the boundary arc of \( A_{d+0.5}(M) \), then the value of \( s_k \) also has nothing to do with \( d \). It is not difficult to figure out that the area of \( P_k \) is \( \frac{k-1}{4} \tan \left( \frac{\pi}{k} \right) \) for \( 3 \leq k \leq 6 \). And the results are as follows: \( s_3 \geq s_4 \geq s_5 \geq s_6 \leq s_7 \leq s_8 \) and \( s_3 = 1.299, s_4 = 1, s_5 = 0.9082, s_6 = 0.8661 \).

With mathematic deduction, we can figure out the values of \( \text{Area}(A_k) \) and \( \text{Area}(E_k) \) and the results are as follows:

\[ \text{Area}(A_k) = \frac{k-1}{4} \tan \left( \frac{\pi}{k} \right) - \left( \sin \frac{\phi}{2} + \frac{\sin^2(\phi/2)}{\tan(\theta/2)} \right) (d + \frac{1}{2})^2 \]  
(2)

\[ \text{Area}(E_k) = \frac{k}{4} \tan \left( \frac{\pi}{k} \right) - \left( \sin \phi - \phi + \frac{\sin^2 \frac{\phi}{2}}{\tan \frac{\theta}{2} (d + \frac{1}{2})} \right) (d + \frac{1}{2})^2 \]  
(3)

According to Eqn. (2) and Eqn. (3), by numerical methods, we can conclude that \( s_i' = \text{Area}(E_i) \) for \( 3 \leq i \leq 6 \) and \( s_i' = \text{Area}(A_7) \). Moreover,

\[ s_3' \geq s_4' \geq s_5' \geq s_6' \leq s_7' \leq \ldots \]

Specially, when applying \( d = 1 \) to Eqn. (2) and Eqn. (3), we can get the corresponding values of \( s_k' \) and we denote them as \( s_k' \). The results are as follows.

\[ s_{31}' = 1.1781, s_{41}' = 0.9717, s_{51}' = 0.8968, s_{61}' = 0.8601, s_{71}' = 0.8550. \]

On the other side, we observe that the curvature of those boundary arcs in Fig. 5 will decrease with the increasing of \( d \). Thus, the area that is cut off by boundary arc from original regular polygon will also decrease. Consequently, the values of \( \text{Area}(E_i) \) and \( \text{Area}(A_k) \) increase with the increasing of \( d \). Therefore, we have \( s_k' \geq s_k' \) for \( 3 \leq k \leq 7 \).

Next, with Euler’s formula we can obtain a better upper bound for \( \alpha(G) \) just as what Du et al. did in [22]. Let \( m \) and \( n \) respectively be the number of edges and vertices in the Voronoi diagram. Let \( f_i \) and \( f'_i \) be the the number of non-boundary Voronoi cells with \( i \) edges and boundary Voronoi cells with \( i \) edges respectively. By Euler’s formula, we have

\[ 1 + \sum_{i=1}^{\infty} (f_i + f'_i) + n = m + 2. \]

Considering the degree of each vertex in Voronoi diagram is at least 3, we have \( 3m \leq 2m \). Then we have

\[ \sum_{i=3}^{\infty} (f_i + f'_i) \geq 1 + \frac{m}{3}. \]  
(4)

It is clear that each boundary Voronoi cell has at least one edge which is on the boundary of \( A_{d+0.5}(M) \). Let \( f_{out} \) denote the total number of those edges in the Voronoi diagram. Then

\[ 2m = f_{out} + \sum_{i=3}^{\infty} i(f_i + f'_i) \geq \sum_{i=3}^{\infty} (i f_i + (i + 1) f'_i). \]  
(5)

Combining Inequalities (4) and (5), we have

\[ \sum_{i=3}^{\infty} (s_i f_i + s'_i f'_i) \leq A_{d+0.5}(M), \]

Considering \( \sum_{i=3}^{\infty} (s_i f_i + s'_i f'_i) \leq A_{d+0.5}(M) \), we have

\[ \sum_{i=3}^{\infty} ((s_i - (s_6 - s_6'))(6 - i)) f_i + (s'_i - (s_6 - s_6'))(5 - i) f'_i \leq A_{d+0.5}(M) - 6(s_6 - s_6'). \]

Noting that \( s_6 > s_6' \), for \( i \geq 7 \), we have

\[ s_i - (s_6 - s_6')(6 - i) \geq s_i \geq s_6 \]

and

\[ s'_i - (s_6 - s_6')(5 - i) \geq s'_i \geq 2(s_6 - s_6') \geq s_6. \]

For \( 3 \leq i \leq 6 \), we can verify that

\[ s_i - (s_6 - s_6')(6 - i) \geq s_i \geq (s_6 - s_6')(6 - i) \geq s_6. \]
and
\[
s'_i - (s_6 - s'_i)(5 - i) \geq s'_{i1} - (s_6 - s'_{i1}) \geq s_6.
\]
Then,
\[
\alpha(G) = \sum_{s=3}^{\infty}(f_i + f'_i) \leq \frac{A_{d+0.5}(M) - 6(s_6 - s'_i)}{s_6}
\]
When \(d \to \infty\), we have \(s_6 - s'_i \to \frac{7\sqrt{3}}{16} = 1\). Hence,
\[
\alpha(G) \leq \frac{A_{d+0.5}(M)}{2} - 0.0718 \tag{6}
\]

B. A New Upper Bound
Up to now, actually we did not make full use of the property of \(d\)-MIS. What we used for our analysis in Sec. V-A is just that for any two nodes in \(d\)-MIS, their Euclidean distance is greater than 1. In this subsection, we will analyze the properties of \(d\)-MIS more specifically so that we can get a better upper bound for \(\alpha(G)\).

In [29], Zhang et al. proposed a novel idea to analyze the upper bound of \(\alpha(G)\). And they gave \(\alpha(G) \leq \beta \gamma(G)\), where \(\beta\) refers to how many independent nodes can be contained in \(N^d(\alpha)\) for any node \(\alpha\). The expression of \(\beta\) is
\[
\beta = \begin{cases}
5, & \text{if } d = 1, \\
21, & \text{if } d = 2, \\
5 + \frac{4d(d + 1)}{2}, & \text{if } d \geq 3.
\end{cases}
\]
This result improved the previous upper bound from \(O(d^2)\) into \(O(d)\). Nevertheless, this result can still be greatly improved and we will show it as follows.

We divide the whole \(d\)-hop independent nodes \(I\) into two classes. They are
\[
A = N^{[d/2]}(M) \cap I = \{u_1, u_2, \ldots, u_t\},
\]
and
\[
B = N^{d/2}(M)\setminus N^{[d/2]}(M) \cap I = \{v_1, v_2, \ldots, v_s\}.
\]
For the first class, we are to show that \(t \leq 3.399\gamma(G) + 4.874\).

For \(1 \leq i \leq t\), denote the shortest path from \(u_i\) to \(M\) as \(P_i\) and let \(w_i\) be the last node on \(P_i\) before reaching \(M\). Because \(N^1(M)\) contains at most \(3.399\gamma(G) + 4.874\) 1-hop independent nodes [22], if \(t > 3.399\gamma(G) + 4.874\), we could always choose two nodes \(u_i\) and \(u_j\) such that \(w_i\) and \(w_j\) are adjacent. Then, \(u_iP_iw_iP_ju_j\) is a path from \(u_i\) to \(u_j\) with length of at most \(2[(d/2) - 1] + 1 \leq d\), which contradicts that \(u_i\) and \(u_j\) are \(d\)-hop independent. Here, \(P_j\) is the reverse of \(P_j\).

Next, we discuss the rest of \(d\)-hop independent nodes in \(I\). For \(1 \leq i \leq s\), denote the shortest path from \(v_i\) to \(M\) as \(Q_i\) and let \(S_i\) be the set of nodes which are in \(N^{(d-1)/2}(v_i)\cap Q_i\). Obviously, for each \(Q_i\), its length is at least \(\frac{d}{2} + 1\). Consider any two nodes \(v_i, v_j \in B\). We are to show that any node in \(S_i\) is not adjacent to any node in \(S_j\). Suppose there are two nodes \(w \in S_i, z \in S_j\) and \(w\) is adjacent to \(z\). Then, there exists a path from \(v_i\) to \(v_j\) which contains \(w\) and \(z\), and the length of this path is at most \(2[(d - 1)/2] + 1 \leq d\), which contradicts that \(v_i\) and \(v_j\) are \(d\)-hop independent. Hence, \(A_{0.5}(S_i)\) does not intersect with \(A_{0.5}(S_j)\). Since \(Q_i\) is the shortest path from \(v_i\) to \(M\), any node in \(Q_i\) is not adjacent to its second successor. Thus, \(S_i\) contains at least \(\frac{1}{2}[(d - 1)/2]\) disjointed nodes which also means they are 1-hop independent. That is to say, one \(d\)-hop independent node corresponds to \(\frac{1}{2}[(d - 1)/2]\) 1-hop independent nodes. Moreover, for any node \(w \in S_i\), the hop distance from \(w\) to \(M\) is greater than \([d/2] + 1\) - \([(d - 1)/2]\) ≥ 1, which means \(w\) is not in \(A_{0.5}(M)\).

According to the result in Sec. V-A, we have
\[
s \leq \frac{A_{d+0.5}(M) - A_{0.5}(M)}{2} - 0.0718
\]
Combining the Inequality (1) and \(A_{0.5}(M) \geq \pi/4\), we have
\[
s \leq \frac{\pi(d + \frac{1}{2})^2 + (\gamma(G) - 1)\Delta S - 0.8572}{\frac{\sqrt{3}}{2} - \frac{1}{2} \frac{d - 1}{2}}.
\]
Consequently, we obtain a new upper bound as follows:
\[
\alpha(G) \leq \begin{cases}
\frac{\pi(d + \frac{1}{2})^2 + (\gamma(G) - 1)\Delta S - 0.0718}{\frac{\sqrt{3}}{2} - \frac{1}{2} \frac{d - 1}{2}}, & \text{if } d \leq 2, \\
\frac{\frac{\sqrt{3}}{2} + \frac{1}{2} \frac{d - 1}{2}}{\Delta S}, & \text{if } d \geq 3.
\end{cases}
\]
Let \(\lambda\) be the ratio of \(\alpha(G)\) and \(\gamma(G)\). Then, we have
\[
\lambda = \begin{cases}
\frac{\Delta S}{\frac{\sqrt{3}}{2} - \frac{1}{2} \frac{d - 1}{2}}, & \text{if } d \leq 2, \\
\frac{\Delta S}{\frac{\sqrt{3}}{2} + \frac{1}{2} \frac{d - 1}{2}}, & \text{if } d \geq 3.
\end{cases}
\]
With numerical method, we can conclude that the value of \(\lambda\) is maximum when \(d = 6\) and is equal to 18.4. Moreover, when \(d \to \infty\), \(\lambda \to 12.6366\). Therefore, for arbitrary value of \(d\), the value of \(\lambda\) is no more than 18.4. In this sense, \(\lambda\) has little relation to \(d\). Therefore, our analysis improve the previous \(O(d)\) into \(O(1)\). This is a huge improvement in the related area.

VI. Performance Analysis
In this section, we discuss the approximation ratio of CS-Cluster. Assume the size of an optimal \(d\)-MCDS is \(opt\). Through Alg. 1, we get a \(d\)-MIS, namely the set \(M\). According to the analysis in Sec. V, we have \(|M| \leq \lambda opt\). As for Alg. 2, we have Lemma 4 below.

**Lemma 4.** The total number of connectors selected in Alg. 2, namely \(C_{\text{\lambda}}\), is at most \(2d\lambda opt\).

**Proof:** In the first round, as we described above, Alg. 2 is to connect all \((r(1)-1)\)-hop connected components into one \((r(1)-1)\)-hop connected component. Moreover, before the end of the current round, there always exists one node that can \((r(1)-1)\)-hop connect two \((r(1)-1)\)-hop connected components. Let \(v\) be that
node. Then we have
\[ \text{cost}(v) \geq \frac{1}{r(0) - 1} = \frac{1}{2d}. \]

Since Alg. 2 always selects the node with the largest value in each iteration, the node that is chosen in this iteration has the value of at least \( \text{cost}(v) \). 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 \( \text{comp}(v) \). We select all the nodes in the shortest paths from \( v \) to the \( \text{comp}(v) \) \( r(1) \)-hop connected components. Obviously, all those chosen nodes can connect \( M \) into a d-CDS. Besides, the size of \( C_1 \) 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. 2 begins. Thus, the size of these connectors is
\[ \sum_{v \in C_1} \text{no.}(v) \leq \sum_{v \in C_1} \frac{\text{comp}(v)}{\text{cost}(v)} \leq 2d|M|. \]

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
\[ |M| + \sum_{j=1}^{i-1} |C_j| \leq 2^{i-1}|M|. \]

There also exists an alternative algorithm to connect \( M \cup_{j=1}^{i-1} C_i \) into a d-CDS, and the size of the corresponding connectors is
\[ \sum_{v \in C_i} \text{no.}(v) \leq \sum_{v \in C_i} \frac{\text{comp}(v)}{\text{cost}(v)} \leq (r(i - 1) - 1)2^{i-1}|M|. \]

Since
\[ (r(i - 1) - 1)2^{i-1} = \left( \frac{r(i - 2) + 1}{2} \right) \cdot 2 \cdot 2^{i-2} \leq r(i - 2) - 1 \cdot 2^{i-2} \leq \ldots \leq r(0) - 1 = 2d, \]
we have
\[ \sum_{v \in C_i} \text{no.}(v) \leq 2d|M|. \]

Finally, in the \( i^{th} \) round where \( r(t) = 1 \), Alg. 2 does the same with its corresponding alternative algorithm. Therefore, \( |C \setminus M| = \sum_{v \in C_i} \text{no.}(v) \leq 2d|M| \), and Lemma 4 follows.

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

**Corollary 1.** After the \( i^{th} \) round, Alg. 2 will indicate a feasible solution \( F_i \) with size \( \sum_{v \in C_i} \text{no.}(v) \) 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 \((2d + 1)/\lambda\).  

## VII. Simulation

In this section, we compare CS-Cluster with previous literature [28] for the same problem (refer as Gao’s algorithm), which is the most related work for d-MCDS problem. 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.

Fig. 6(a)-(c) exhibit the performance comparisons between two algorithms with different \( d \). From the figure, we can conclude that CS-Cluster’s always performs better than Gao’s algorithm under different settings. Moreover, when \( d \) increases, the superiority of CS-Cluster also increases. Considering the great influence of Alg. 3, we also compare the performance of CS-Cluster without Alg. 3 with Gao’s algorithm. From Fig. 6, we find that CS-Cluster without Alg. 3 still produces

![Fig. 6](image-url)  
(a) Comparison when \( d = 2 \)  
(b) Comparison when \( d = 3 \)  
(c) Comparison when \( d = 4 \)

Fig. 6. Performance comparison between CS-Cluster and Gao’s algorithm in [28] in different cases where \( d = 2, 3, 4 \) respectively.

Fig. 7. A sample of UDG which contains 122 nodes. Gao’s algorithm selects a 2-CDS with 49 nodes while CS-Cluster selects a 2-CDS with 36 nodes.
smaller $d$-CDS than Gao’s algorithm. We think that the great advantage of CS-Cluster comes to the sparsity of $d$-MIS by Alg. 1, the flexibility of Alg. 2, and the feature of non-redundancy by Alg. 3.

Fig. 7 shows a sample comparison between two algorithms with 112 input nodes where $d = 2$. We can clearly see the superiority of CS-Cluster since Gao’s algorithm constructs a 2-CDS with 49 nodes while CS-Cluster only uses 36 nodes.

VIII. Conclusion

In this paper, we studied the multi-hop connected clustering problem for a given wireless sensor 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 $\lambda$ for the dominator size in a unit disk graph, and proved that $\lambda$ is no more than 18.4. As a result, we reduce the bound of $O(d)$ from previous literature [29] 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. In all, CS-Cluster is flexible to obtain a nearly optimal $d$-MCDS and is suitable for distributed environments. Our simulation results also exhibited the outstanding performance of CS-Cluster.

Acknowledgement

This work has been supported in part by the National Natural Science Foundation of China (Grant number 61202024, 61472252, 61133006, 61422028), China 973 project (2012CB316201), Shanghai Educational Development Foundation (Chenguang Grant No.12CG09), Shanghai Pujiang Program 13PJ1403900, the Natural Science Foundation of Shanghai (Grant No.12ZR1445000), and in part by Jiangsu Future Network Research Project No. BY2013095-1-10 and CCF-Tencent Open Fund.

References