Approximation Algorithms for the Connected Dominating Set Problem in Unit Disk Graphs

Gang Lu, Ming-Tian Zhou, Yong Tang, Ming-Yuan Zhao, Xin-Zheng Niu, and Kun She

Abstract—The connected dominating set (CDS) problem, which consists of finding a smallest connected dominating set for graphs is an NP-hard problem in the unit disk graphs (UDGs). This paper focuses on the CDS problem in wireless networks. Investigation of some properties of independent set (IS) in UDGs shows that geometric features of nodes distribution like angle and area can be used to design efficient heuristics for the approximation algorithms. Several constant factor approximation algorithms are presented for the CDS problem in UDGs. Simulation results show that the proposed algorithms perform better than some known ones.

Index Terms—Approximation algorithm, connected dominating set, unit disk graph.

1. Introduction

A dominating set (DS) of a graph $G=(V, E)$ is a subset of nodes $DS \subseteq V$ such that every node of $G$ is either in DS or adjacent to at least one node in DS. If the subgraph induced by DS (denoted by $G[DS]$) is connected, we call this DS the connected dominating set (CDS). The CDS problem asks for finding a CDS of a graph with its size as small as possible (i.e., minimum CDS or MCDS). However, finding MCDS for arbitrary graphs has shown to be an NP-complete problem, thus we can not expect for an optimal solution in polynomial time unless $P=NP$.

Wireless networks, such as sensor networks and ad hoc networks, due to their flexibility, have been widely applied in the fields like military communications, emergence systems, e-commerce, etc. The initial topology of those networks can be modeled as the unit disk graph (UDG) which consists of a node set $V$ and an edge set $E$ such that any edge $(u, v)$ belongs to $E$ if and only if the distance between $u$ and $v$ is no more than $r$, where $r$ means the transmission range of the nodes and all nodes have the same $r$. For the purpose of efficient communication, it is necessary to set up an MCDS as virtue backbone. However, even in UDG, the problem of finding an MCDS is still NP-complete. Thus, especially in practice, our only chance is to find a near optimal one through some approximation algorithms. To our knowledge, most of such approximations are designed with the consideration only focusing on the logical relationships between nodes, for example, degrees or links. The geometric properties are usually used in establishing some theoretical bounds. In this paper, we show that some geometric properties, such as angle and area, can be used as efficient heuristics when designing algorithms.

We find that though some known algorithms possess the tight constant approximation factor, the actual sizes of CDS constructed by those algorithms are rather large (see Section 4.5), which may cause more energy consumption and suffer more time delay when forwarding packets. This fact implies that the current approximation factors do not reflect the actual performance of the algorithm outputs, and some times, especially in practice, we may not need to worry about this parameter too much. In addition, some worst case (time or message) complexities, though have theoretical means, do not reflect the actual implementation cost of the algorithms. In other words, the average case complexities (See section 3) are the more practical metrics that should be taken into consideration in wireless scenarios.

The rest of the paper is organized as follows. Section 2 introduces the related works. Section 3 deals with notations, definitions and some assumptions. Section 4 presents our algorithms and some theoretical results. Section 5 shows the simulation results. Section 6 concludes the paper.

2. Related Works

In early of 1960s, Claude Berge first formulated the mathematical concept of domination in graphs. Half century has passed since then and thousands research papers have been published on this topic.

Guha et al. proposed two centralized algorithms for CDS problem in undirected graphs in [4]. Using polynomial time, their algorithms can achieve approximation ratios of $2H(\Delta)+2$ and $H(\Delta)+2$, where $H(\Delta)$ is the harmonic number of degree $\Delta$. The tight constant approximation factor is no more than $\Delta$.
respectively, where $\Delta$ is the maximum node degree and $H$ is the harmonic function. Though ten years passed, Guha’s first growing algorithm is still one of the most efficient algorithms for CDS problem in practice. Wan$^{(3)}$ et al. proposed an approximation scheme with a ratio of 8 for CDS problem in UDGs. Their scheme first finds a maximal independent set (mIS, an independent set which is also a DS) and then connects it. In $[6]$, Funke$^{(4)}$ et al. presented a simple distributed approximation algorithm for CDS problem in UDGs. The main contribution of $[6]$ is that the authors proved that the size of any mIS (denoted by $|\text{mIS}|$) is at most $3.453 \times |\text{OPT}| + 8.291$, where $|\text{OPT}|$ denotes the cardinality of optimal solution (i.e. $|\text{MCDS}|$). Two years later, through a careful analysis, Vahdatpour$^{(5)}$ et al. enhanced the bound to $|\text{mIS}| \leq 3|\text{OPT}| + 3$ and showed that the multiplier 3 in the above formula is tight. Note that with this tight bound, a distributed connection algorithm can achieve the approximation factor of 6 (times $|\text{MCDS}|$). Though possessing constant approximation ratio, the “mIS+connection” styled algorithms perform poor in practice. We will discuss some drawbacks of such styled algorithms in Section 4.5. Recently, Shang$^{(6)}$ et al. studied the $m$-connected $k$-dominating set problem, which is an important extension for the traditional CDS problem in UDGs and the algorithms for the $m=2$ case are obtained.

Exact algorithms for MDS (minimum dominating set), MIS (maximum independent set), and MCDS problems can be found in $[9]$, $[10]$, and $[11]$, respectively. According to our experimental results, these algorithms are the most efficient ones obtained so far.

3. Preliminaries

Besides the terms and conceptions introduced in the above contents, we use the following notations, definitions, and assumptions to aid our presentation.

1) $K$, $M$: $K$ is a 2-D network region. Unless special noted, we assume $K$ is a square with side length $M$. In some formulas, if not confused, we also refer $K$ to denote the area of network region and we usually assume $K >> \pi r^2$, where $r$ is the transmission range of nodes.

2) $n$ is the number of nodes deployed in $K$, and throughout this paper we assume all nodes uniformly and independently distributed in $K$.

3) $\lambda$ denotes node density, i.e. $\lambda = n/M^2$.

4) $h$ is the unit distance and defined as $h = M/\sqrt{n}$.

5) $|S|$ denotes the number of elements in set $S$.

6) Minimal dominating set (mDS) is a DS such that any proper subset of mDS is not a DS; in other words, for any $s \in \text{mDS}$ there exists at least one node $u \in V - \text{mDS}$ such that $u$ is not dominated by any nodes in mDS except $s$.

7) Minimal connected dominating set (mCDS) is a CDS such that removing any node from this set will make it no longer a CDS.

8) $N^r(u)$ is a set of nodes of which the distance to $u$ is exactly $t \geq 0$ hops ($u$ might also be a set of nodes).

9) $N^t[u] = N^t(u) \cup u$.

In this paper, unless specially noted, we assume both $\lambda$ and $r$ are within some positive constants (i.e. the average case). Clearly, this assumption is reasonable in wireless scenarios. The complexity obtained under such assumption is called the average complexity which reflects the practical cost of the implementation.

4. Main Results

4.1 Connectivity of UDG

Theorem 1. Let $n$ nodes be uniformly and randomly distributed in $K$ and $n$ is large, if we want the probability $\Pr \{\text{UDG be connected}\} > \beta$, where $0 \leq \beta \leq 1$, the transmission range $r$ should satisfy

$$r > 2h \left( \frac{\ln(1 - \frac{1}{4\beta})}{\pi} \right).$$

Proof. The probability for a node, say $u$, finding its nearest neighbor in $u$’s neighbor region $\delta$, can be written as

$$\Pr = 1 - \left(\frac{K - \delta}{K}\right)^{n-1} \approx 1 - \exp(\lambda \delta).$$

Now let’s consider a virtual incremental construction which connects all nodes in $K$. At each step, we add a node which is the nearest one to the previous connected component to form a larger component (can be seen as a super-node). The initial component is an arbitrary node, and the process will come to end when no isolated node remains. Note that the worst case for a node to seek its nearest neighbor occurs at the corner of $K$, i.e. $\delta = \pi r^2/4$, we have

$$\Pr \{\text{UDG connected}\} \approx \left(1 - \exp \left( -\frac{n}{M^2} \left(\frac{\pi r^2}{4}\right)^{n-1} \right) \right) > \beta.$$ 

Noting that $h = M/\sqrt{n}$, we can obtain the desired result through some manipulations.

For example, if there are 100 nodes in $K$, then choosing $r = 3.5$ h will make the UDG be connected with the probability more than 99%.

4.2 Some Properties of IS in UDG

Independent set (IS) plays an important role in both the construction and performance analysis for CDS in UDGs.

Theorem 2. In UDG, the following properties hold: (a) $|\text{MDS}| \leq \frac{2\sqrt{3}K}{3r^2}$, (b) there exists a $|\text{CDS}| \leq \frac{4\sqrt{3}K}{3r^2} - 2$, and (c) there exists a $x \in \left[1, \frac{2\sqrt{3}K - 12r^2}{9r^2}\right]$, which
satisfies $|\text{MCDS}| \geq x$.

Proof. 1) L. Fejes Tóth in 1950s showed that the densest packing of unit disks can be achieved through a hexagonal lattice. Under our model of networks the area of each grid can be calculated as $\sqrt{3}r^2/2$. Thus, any IS including MIS can contain at most $2\sqrt{3}K/3r^2 - 1$ nodes.

Noting that an MIS must be a DS, the property (a) holds. 2) Through the property (a), we can obtain an mIS with the shortest distance of any node, say $u$, in this mIS to $\{\text{mIS} - u\}$, is within two hops. We can connect this mIS to form a CDS using at most $|\text{mIS}|$ nodes; 3) The property (c) is followed by the result of property (a) and the upper bound obtained in [7] (see Section 2).

Proposition 1. Let $m$ be an integer and $|\text{IS}|$ be the average size of IS, we have

$$|\text{IS}| \geq \sum_{m=1}^{n} \left(1 - \frac{\pi r^2}{K}\right)^m$$

Proof. Because the subsets in $V$ are not independent to each other, it is very difficult to obtain the exact expression of $\Pr(|\text{IS}| = m)$. So, we turn to consider a lower bound of $|\text{IS}|$: given an IS of size $m$, then no edge can exist between any pair of nodes in IS. It follows that

$$\Pr(|\text{IS}| = m) \geq \left(1 - \frac{\pi r^2}{K}\right)^m$$

where $K' = K + \partial K$, $\partial K$ denotes the boundary area of $K$, see Fig. 1 for illustration (shaded region). In case that $K$ is not large as compared to $r$, we should consider boundary effect.

The lower bound of expectation of $|\text{IS}|$ can be computed by (2).

Fig. 1. Illustration for the boundary area of $K$.

Fig. 2. Theoretic and simulation results of $|\text{IS}|$ in UDGs.

Fig. 3. The maximum interval angle $\alpha$ of $u$. 4.3 Centralized Algorithms for mDS and mIS

Starting from any mDS or MIS, we can build a CDS for UDG.

Observation 1. The CDS problem is to find a spanning tree with the maximum number of leaves. To make the number of leaves large, intuitively, we tend to choose nodes near to the “border” region as leaves. Here “border” we mean a virtual dividing line (might be a band) which separates a region from its neighbor regions according to some predefined measures. For example, using node density as a measure, we can find some borders between holes and their neighbor regions. Fortunately, detecting such borders does not need the global knowledge of the nodes distribution.

Definition 1. The maximum interval angle of a node is $u(\alpha)$. Consider we sort $N(u)$ (with respect to $u$) to a counterclockwise sequence, then $\alpha$ is defined as the maximum absolute angle difference among each pair of consecutive nodes in $N(u)$. See Fig. 3 for illustration.

We should point out that the techniques to estimate the direction without localization are available, and such issue is discussed in IEEE antenna and propagation community as the angle-of-arrival (AOA) problem.

Given a threshold value $\gamma$ (e.g., $\pi / 2$), a node whose $\alpha$ is larger than $\gamma$ will be considered as in border region. In Fig. 4, nodes in the border region are colored red and marked with triangle.
The following two centralized algorithms can be implemented in some distributed fashions.

**Algorithm 1. angle_mds**

Input: UDG. Output: an mDS for UDG.

Initialization: Each node, e.g., \( u \), computes its \( \alpha \) and maintains a boolean variable \( \text{leaf} \) and is initialized to \( 0 \). \( u \) also maintains a \( \text{linkList} \), which is initialized to \( N(u) \).

1) Sort \( n \) nodes in descending order according to \( \alpha \);
2) For \( u=1 \) to \( n \) do, // check if it is qualified to be a leaf;
3) If node \( u \)'s \( \text{linkList} \neq \emptyset \), and each of \( u \)'s leaf-neighbor’s \( \text{linkList} \) has the cardinality larger than \( 1/\text{leaf} \) neighbor is a neighbor whose \( \text{leaf}=1 \);
4) \( u \) sets its \( \text{leaf}=1 \) and inform \( N(u) \) to delete \( u \) from their \( \text{linkList} \);
5) Return the nodes whose \( \text{leaf}=0 \) to be an mDS.

**Algorithm 2. star_fomation**

Input: UDG and its mDS. Output: an mIS of UDG.

Initialization: Each node maintains a boolean variable \( \text{leaf} \) which should be initialized to \( 0 \) if this node belongs to the output set of \( \text{mDS} \); otherwise, sets \( \text{leaf}=1 \). Each node also maintains a \( \text{linkList} \) to record all its dominators.

1) Find a connected component \( C (|C|>1) \) in \( G(\text{mDS}) \);
2) If no such \( C \) can be found, return the nodes whose \( \text{leaf}=0 \) as \( \text{mIS} \); else go to the next step.
3) Choose an arbitrary node, say \( v \), from \( C \) and find a subset \( U \subseteq N(u) \) such that all nodes in \( U \) are dominated only by \( u \) and \( |U| \) should be as large as possible. If \( |U|=0 \), go to step 5; else go to the next step.
4) Choose a node \( v \in U \) to set \( v \)'s \( \text{leaf}=0 \) and inform \( N(v) \) to add \( v \) to their \( \text{linkList} \). Set \( U'=U-N(v) \); repeat step 4 until \( U=\emptyset \).
5) Set \( u \)'s \( \text{leaf}=1 \) and inform \( N(u) \) to delete \( u \) from their \( \text{linkList} \), then go to step 1.

Fig. 4. Illustration for the border region: (a) UDG, \( n=100, r=1.9h \) and (b) nodes in the border region.

Fig. 5. Examples for the Algo. 1 and Algo. 2: (a) an mDS constructed by angle_mds and (b) an mIS constructed by star_fomation.

Fig. 5 shows examples for the above two algorithms.

**Theorem 3.** angle_mds and star_fomation can be implemented in \( O(n\log \Delta + n\log n) \) and \( O(n\Delta) \) time, respectively; in the average case, the time complexities are \( O(n\log_2 n) \) and \( O(n) \), respectively.

**Proof.** In angle_mds a node, say \( u \), will use \( O(\Delta \log \Delta) \) time to sort its neighbors into counterclockwise order and then compute \( \alpha \) in \( O(\Delta) \) time. Step 1 will cost \( O(n \log \Delta) \) time. In step 3, \( u \) checks at most \( O(\Delta) \) neighbors. In step 4, \( u \) updates its \( \text{linkList} \) at most \( O(\Delta) \) times when \( N(u) \) inform \( u \) and each can be done in \( O(1) \) time if an adjacency-matrix representation for UDG is employed. So the total time complexity is \( O(n\log \Delta + n\log n) \).

In star_fomation, a node \( u \in \text{mDS} \) will use \( O(\Delta) \) time to find \( U \subseteq N(u) \) in step 3; noting that \( u \) has at most five independent neighbors, step 4 then at most repeats itself 5 times and each consumes at most \( O(\Delta) \) time; step 5 also needs \( O(\Delta) \) time. The size of mDS could be \( O(n) \), thus the total time needed by Algorithm 2 is \( O(n\Delta) \).

The average case in which \( r \) and \( \Delta \) are within some positive constants (i.e., \( \Delta \) is a constant) is the most common situation we meet in practice. Obviously, in such case the time complexities for the above two algorithms should be \( O(n\log n) \) and \( O(n) \), respectively.

Actually, to build up a UDG, \( O(n^2) \) time is needed in the worst case. Thus, no matter what kind of algorithm be employed, to construct the backbone for UDGs, the total time complexity is at least \( O(n^2) \) in the worst case.

**Theorem 4.** The output of angle_mds is an mDS.

**Proof.** First we shall prove the output is a DS. If an arbitrary node \( v \) is a leaf, then \( v \)'s \( \text{linkList} \) can not be empty. Now we assume that \( u \) is the \( v \)'s unique dominator. When the algorithm scans \( u \) (step 2), \( u \) can not set its \( \text{leaf} \) value to \( 1 \) because \( u \) has a leaf \( v \) whose \( |\text{linkList}|=1 \). In other words, \( u \) must be kept in the output set.

Second, we shall prove the output set is also an mDS. Assume the output is not an mDS. There exists a node \( u \) in the output set with \( N[u] \) dominated by other nodes in DS (i.e., \{DS-\u\}). We assert that \( u \)'s \( \text{linkList} \) can not be empty, and each of \( N(u) \)'s \( \text{linkList} \) has the size at least 2. Thus, when algorithm scans \( u \) (step 2), \( u \) can not set its \( \text{leaf} \) value to \( 1 \) because \( u \) has a leaf \( v \) whose \( |\text{linkList}|=1 \). In other words, \( u \) must be kept in the output set.

Theorem 5. The output of star_fomation is an mIS.

**Proof.** Firstly, if there exists a connected component \( C \) with \(|C|>1\), the algorithm will not terminate. However, due to step 3 to step 5, at each iteration, at least one node \( u \) in \( C \) sets its \( \text{leaf}=1 \), the size of \( C \) then decreases by 1 until \(|C|=1 \). In addition, step 4 guaranteed that the selected node \( v \in U \)
is not adjacent to any previously obtained dominators whose leaf=0. In other words, the number of the connected components can not be increased during the procedure. So, finally we will get an output which is an IS. Secondly, this IS is also a DS. Because at each time when deleting a node u from mDS, we also add some nodes belong to U in order to dominate the nodes only dominated by u. Thus, all nodes in G must be dominated unless the input is not a mDS. Thirdly, the IS is also an mIS. Because a node in IS must dominates itself; if we delete this node, this IS will no longer be a DS.

**Theorem 6.** Given an mDS for UDG as input, the mIS constructed by star_fomation satisfies |mIS| ≤ 4|mDS|−3.

**Proof.** Two adjacent nodes, say u and v respectively, can have no more than 8 independent nodes in N(u)∪N(v)[7]. Thus, transforming a dominator from a component O(C(C)>1) to leaf will incur at most 4 additional nodes to be turned from leaf to dominator. In addition, we need at most transform |mDS|−1 nodes in mDS from dominator to leaf. So we have, |mIS| ≤ 4(|mDS|−1)+1+4|mDS|−3.

**Theorem 7.** Starting from an mIS constructed by “Algorithm 1+Algorithm 2”, we can build a CDS with its size no more than 9|MCDS|+9.

**Proof.** To connect any node u∈mIS with its complementary set {mIS−u}, it needs to pick up one or two nodes from {V−mIS}. Noting that |MIS| ≤ 3 |MCDS| +3[7], we can complete the proof.

### 4.4 Distributed Algorithm for mIS

In this subsection, we assume the operation of algorithm is divided into rounds, and at each round all nodes are assigned a time slot to perform some actions. The transition of the conditions will cause no delay in time. We only sketch the main ideas of the algorithm, and some techniques presented in [12] and [13] may help us to reduce the consumed time, and we will deal with such issue in our future works.

**Algorithm 3. t_angle mIS**

Input: a UDG, t (a positive integer). Output: an mIS.

Initialization: Each node computes its α value and maintains two local variable: a boolean variable leaf which is initialized to 0, and a boolean variable active, which is initialized to 1, to indicate node’s condition. Each node (e.g. u) also maintains a data structure smallerList which records the nodes within t hops whose α values smaller than that of u.

1) At each round scan all nodes that are active (active=1), other nodes just keep silent and wait for their time slot, if no such node exists (time has expired), return the nodes whose leaf=0 as mIS; else go to the next step.

2) If the predefined (round) time limitation achieves, go to 1). Otherwise, assume now is in u’s time slot. If any node in u’s smallerList is active, u keeps silent in its time slot and algorithm goes to step 2; else go to the next step.

3) All the active nodes in N(u) set their leaf=1 and active=0, and inform active nodes within t hops their decisions; u sets its active=0 and informs the active nodes within t hops(to u) its decision; go to step 2.

**Theorem 8.** The output of t_angle_mIS is indeed an mIS.

The correctness of Theorem 8 can be easily verified.

**Theorem 9.** Let t (>0) be a constant, the message complexity of t_angle mIS is O(n log n+nf(t)), where f(t) = max u∈U N(u), and 0 ≤ f(t) ≤ n. In the average case, the message complexity required by the major routine of t_angle_mIS is O(n).

**Proof.** First, to assign time slot to each node, we need to construct a spanning tree of G. Using techniques presented in [13], we can do this with O(n log n) message complexity, and it is optimal. Second, to compute α of each node (e.g. u), u should exchange packets with N(u) and its message cost is O(f(l) = Δ). Third, in Algorithm 3 a node u only needs to exchange messages with neighbors in t hops twice: one is in the initialization and the other is in step 3 when u sets active=0. Note that any node changes its condition only once over the entire procedure, u then sends response to the informing message at most O(f(t)) times. So, the total message complexity is O(n log n+nf(t)). For the average case, when t and Δ both are constants, f(t) should be also within a constant.

Thus, the average message complexity for major routine (i.e. initialization plus the steps 1 to 3) of t_angle_mIS is O(n).

From above content we can see that the time slots needed by Algorithm 3 are O(n^t) . Some techniques introduced in [12] and [13] may help us to reduce the consumed time, and we will deal with such issue in our future works.

We can connect an mIS to form a CDS for UDG, and the factor is guaranteed as shown in Theorem 7.

### 4.5 Redundancy in CDS Constructed by mIS+Connection Styled Algorithms

**Observation 2:** 1) though possessing the tight constant approximation factor, the actual performance of CDS constructed by mIS+Connection styled algorithms (see Fig. 6 (b) to Fig. 6 (d)) may be rather poor as compared to the centralized schemes’ (see Fig. 6 (e) and Fig. 6 (f)); 2) low constant factor algorithms (see Fig. 6 (b) and Fig. 6 (c)) actually do not show better performance than large constant factor algorithm (see Fig. 6 (d)) as we expected. The reasons for above facts can be interpreted in spirit as follows.

Given a partition of K by an mIS, we can image K has almost been covered once. To connect an mIS, an algorithm should pick up at most another 2|mIS| nodes
from $G$, which can be viewed as we covered $K$ once again. In centralized schemes, for example guided by some greedy strategies, many places of $K$ may have chance that does not need to be covered twice. The performance of mIS+Connection styled algorithms then may be rather low as compared to the centralized schemes, and we can expect that many nodes of CDS constructed by such styled algorithms are not necessary.

We define dense_mIS an mIS in which the shortest distance between one node and other nodes of this mIS is exactly two hops (see Fig. 6 (a)), and sparse_mIS an mIS which is not a dense mIS.

According to the properties of mIS and the theoretical bound\[^7\] for the relationship between $|\text{MIS}|$ and $|\text{MCDS}|$, dense_mIS+Connection styled algorithms can obtain the factor as 6 whereas sparse_mIS+Connection fashioned algorithm’s factor is 9 (see Theorem 7). However, the factors here do not reflect the real size deference between the CDS constructed from the mIS of different classes. See Fig. 7.

**Algorithm 4. t_Redundancy_Removing**

Input: a CDS, and a constant $t$ (usual value is 1 or 2).
Output: a CDS with some nodes be removed.

Iteration: for each node $u$ of CDS (in $u$’s time slot), check if \{ belongs to a connected component in $G$. If so, removes $u$ from CDS.

The correctness of Algorithm 4 is straightforward. A node $u$ in $t$_Redundancy_Removing only needs to communicate with its neighbors in $t$-hops once when $u$ finds itself be redundant, thus the total message complexity is $O(nf(t))$, where $f(t)$ has been defined in Theorem 9 (In fact, when $f(t) \to n$, $|\text{CDS}|$ is rather small). In the average case, the message complexity of Algorithm 4 itself is $O(n)$.

4.6 Centralized Algorithm for CDS Problem in UDGs

Centralized scheme is a feasible choice when we do have the global knowledge of networks and the distributed implementation is not a crucial requirement which must be met.

**Algorithm 5. greedy_area // grow a spanning a tree.**

Input: a UDG, and a root node. Output: an mCDS.

Initialization: Each node, e.g. $u$, records $N(u)$ and $N_u$ in its local data structure, and colors itself white. Root colors itself black and $N(\text{root})$ color themselves gray.

1) Choose a gray node (e.g. $u$) which dominates the greatest number of white nodes to color it black and the white nodes in $N(u)$ gray. If there are many such gray nodes, choose the (arbitrary) one which covers the largest newly covered area.

The distance for any node and its complementary set in dense mIS (Fig. 7 (b)) is exactly two hops, and sparse mIS (Fig. 7 (a)) may be two or three hops. The two kinds of mIS both cover $K$ once, thus we can expect the nodes density (i.e $|\text{nodes}|/K$) of sparse mIS is no more than that of dense mIS, i.e. $|\text{sparse mIS}| \leq |\text{dense mIS}|$. When we compare the final results of $|\text{CDS}|$, the advantage of dense mIS (less linking nodes will be chosen as compared to that of sparse mIS) over sparse mIS will be diminished by the larger number of the self contained nodes. However, the approximation factors are computed from the process in which both two kinds of mIS are seen as MIS, and the difference of the sets size is not considered. This is the reason why the better factor possessing algorithms do not show better performance as expected. We design algorithm 4 to remove some redundant nodes from CDS.

Fig. 6. Illustration for CDS constructed by different algorithms: (a) Wan’s algorithm[^3], phase 1, (b) Wan’s algorithm[^3], phase 2, (c) Funke’s[^6] algorithm, (d) t_angle_mIS ($t=2$)+Connection, (e) Guha’s algorithm[^4], and (f) greedy_area (= optimal[^11]).

Fig. 7. Two different classes of mIS: (a) sparse mIS and (b) dense mIS.

Fig. 8. illustration for the newly covered area of disk $i$. 

2) Repeat step 1 until no white node remains.
3) (Optional: sort the black nodes in descending order according to their \( \alpha \) value.) Check each black node whether can be removed with the complementary black nodes remain a CDS. If so, we color this black node gray and turn to the next one.
4) Return the black nodes as the \( m \)CDS.

Newly covered area (\( A_{\text{new}} \)) consider now we want to color a gray node to be the \( n \)th (\( \geq 1 \)) black node, then \( A_{\text{new}} \) can be defined as (see Fig. 8, shaded region):

\[
A_{\text{new}} = \left( \sum_{j=1}^{n} \text{disk}_j \right) - \left( \sum_{j=1}^{n-1} \text{disk}_j \right)
\]

(4)

where \( |\text{disk}_j| = \pi r^2 \), and centered at (a black node) \( j \).

\( A_{\text{new}} \) does not need to be calculated precisely, and some approximation methods can be used. To compute \( A_{\text{new}} \) of a node \( u \), we only need to consider the black nodes with the distance to \( u \) no more than \( 2r \), i.e. \( N^2(u) \).

**Theorem 10.** The output of greedy\_area is an mCDS.

This property is guaranteed by step 3.

**Theorem 11.** The time complexity of greedy\_area is at most \( O(nm) \), where \( m \) is the number of all edges.

**Proof.** It is not difficult to see that the time complexity of Algorithm 5 is dominated by step 3. The optimal solution of CDS may has \( O(n) \) vertices, therefore, in step 3, we need to check \( O(n) \) nodes and each consuming at most \( O(m) \) time to find the redundancy.

Examples for Guha’s centralized algorithm and greedy\_area have been shown in Fig. 6 (f) and Fig. 6 (e) respectively, and they both have the \( O(nm) \) time complexity.

**Proposition 2.** greedy\_area is a constant factor approximation algorithm for CDS problem in UDGs, i.e. \( |\text{mCDS}| \leq c |\text{MCDS}| \), where \( c \) is a positive constant.

We show the reasons for the preceding proposition in spirit as follows: let \( u \) be a node belongs to MCDS, and the set of nodes in \( V \) dominated by \( u \) is called \( S_u \). Obviously, the subsets \( \{ S_{\text{neMCDS}} \} \) form a partition of nodes in \( V \). (If a node is dominated by more than one node in MCDS, we arbitrarily put it in one of \( S_{\text{neMCDS}} \). If we can show that the number of nodes chosen by greedy\_area from any \( S_u \) is within a constant \( c \), proposition 2 then follows.

A node, say \( u \), belonging to an mCDS is either for the purpose of domination or for connection (or for both). See the geometry division of \( S_u \) as shown in Fig. 9, we assert that for the purpose of dominating and connecting all nodes in \( S_u \), greedy\_area needs to pick up at most twelve nodes from \( N(u) \) given that \( u \notin \text{mCDS} \) (if \( u \in \text{mCDS} \), then \( S_u \) is connected and dominated; and redundancy nodes will be removed according to step 3).

![Fig. 9. A cone partition of disk\(_u\).](image)

Fig. 9. A cone partition of disk\(_u\). Fig. 10. the densest packing.

![Fig. 10. the densest packing.](image)

Fig. 10. the densest packing.

We now turn to consider the connection and domination for the outside of \( S_u \). According to the geometric property as shown in Fig. 10, actually, we can place at most twelve independent nodes in \( N^2(u) \). See Fig. 11, let \( v \) be a node lying on the circle centered at \( u \) with radius \( 2r \). Without loss of generality, we assume that there are some white nodes which also lies on the circle and are adjacent to \( v \). If the number of such nodes exceeds 2, instead of bothering some nodes in \( N(u) \) (which lie on the circle with radius 1), the greedy strategy will guide the algorithm to choose \( v \) to be a black node because in this case \( v \) dominates larger number of white nodes. Combining the above results, we concluded that for domination and connection with \( S_u \) and the outside of \( S_u \), greedy\_area will choose no more than \( c=12+2 \times 12=36 \) nodes from \( S_u \).

5. **Performance Study**

In this section, we compare our algorithms with some known ones through simulations.

In the first group of simulations, a given number of nodes ranging from 50 to 1600 are randomly and uniformly distributed in \( K \) (\( M=100 \) m). The transmitting range is 3h. All the simulation results are obtained by running the algorithms on 100 connected graphs, and the average values are taken as the outputs.

The last column in Table 1 indicates the class of the implemented algorithm (dis. short for distributed, and cen. short for centralized). In the seventh row of Table 1, we show some intervals which are calculated form the Proposition 1 to predict the sizes of CDS: the left endpoint of interval equals \( 2 \times \text{IS} \) (view IS as a dense IS) and the right endpoint equals \( 3 \times \text{IS} \) (view IS as a sparse IS). Observing the numbers shown in the 4th, 5th and 6th rows, we can find that the actual sizes of CDS fit the theoretic
results quite well. The 6th and 10th rows of Table 1 show that for sparse, moderate and dense sensor deployments, in both distributed and centralized situations, our algorithms outperform others.

In the second group of experiments, let $n=400$ nodes distributed in $K$ and $r$ ranged from $2h$ to $15h$. The results of distributed algorithms are shown in Fig. 13, and the results of centralized algorithms, for clarity, are shown in Table 2. The result shows that for both distributed and centralized cases, our algorithms perform better than others.

Table 1: Results for $|CDS|$ in the first group of simulations ($r=3h$)

<table>
<thead>
<tr>
<th>$n$</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>400</th>
<th>800</th>
<th>1600</th>
<th>Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPR-minid</td>
<td>15.5</td>
<td>34.8</td>
<td>76.1</td>
<td>163.1</td>
<td>343.6</td>
<td>699.5</td>
<td>dis.</td>
</tr>
<tr>
<td>MPR-greedy</td>
<td>11.0</td>
<td>24.1</td>
<td>54.6</td>
<td>115.7</td>
<td>241.9</td>
<td>499.9</td>
<td>dis.</td>
</tr>
<tr>
<td>WPJ</td>
<td>7.2</td>
<td>14.8</td>
<td>28.9</td>
<td>58.5</td>
<td>113.5</td>
<td>226.4</td>
<td>dis.</td>
</tr>
<tr>
<td>Funke</td>
<td>8.4</td>
<td>15.6</td>
<td>28.9</td>
<td>56.8</td>
<td>111.5</td>
<td>211.2</td>
<td>dis.</td>
</tr>
<tr>
<td>$t$ _angle_mIS+</td>
<td>5.3</td>
<td>10.9</td>
<td>21.3</td>
<td>42.1</td>
<td>82.8</td>
<td>164.2</td>
<td>dis.</td>
</tr>
<tr>
<td>Algo.4 ($r=2$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposition 1</td>
<td>13.20</td>
<td>20.31</td>
<td>32.48</td>
<td>53.79</td>
<td>90.135</td>
<td>159.238</td>
<td></td>
</tr>
<tr>
<td>Giha 1 [4]</td>
<td>5</td>
<td>9.4</td>
<td>18</td>
<td>33.7</td>
<td>64</td>
<td>126.2</td>
<td>cen.</td>
</tr>
<tr>
<td>Giha 2 [4]</td>
<td>5</td>
<td>9</td>
<td>18</td>
<td>34.5</td>
<td>65</td>
<td>131</td>
<td>cen.</td>
</tr>
<tr>
<td>greedy_area</td>
<td>4.7</td>
<td>8.9</td>
<td>17.2</td>
<td>33.2</td>
<td>62.2</td>
<td>124.9</td>
<td>cen.</td>
</tr>
</tbody>
</table>

Table 2: Results for $|CDS|$ in second group of simulations

<table>
<thead>
<tr>
<th>$r$ ($h$)</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Giha 1 [4]</td>
<td>70.1</td>
<td>19.9</td>
<td>10.4</td>
<td>6.5</td>
<td>4.2</td>
<td>2</td>
<td>1.9</td>
<td>1</td>
</tr>
<tr>
<td>Giha 2 [4]</td>
<td>73</td>
<td>20.2</td>
<td>9.3</td>
<td>6.1</td>
<td>5</td>
<td>2</td>
<td>1.3</td>
<td>1</td>
</tr>
<tr>
<td>greedy_area</td>
<td>69</td>
<td>20</td>
<td>9.1</td>
<td>5.9</td>
<td>4.3</td>
<td>1.7</td>
<td>1.3</td>
<td>1</td>
</tr>
</tbody>
</table>

In the final group of experiments, we randomly deploy some holes as obstructions in $K$. Other parameters are the same as that of the second group simulations. A typical scenario has been illustrated in Fig. 12. The simulation results for distributed and centralized algorithms are shown in Fig. 14 and Table 3, respectively.

Once again, in most simulations, we can see that our algorithms perform better than others. Some other experimental results, for example the comparison with the exact-solutions, are omitted due to space limitation.

6. Remarks and Future Works

The main contributions of this paper are as follows: Theorem 1 reveals the relationship between $r$, $n$ and the probability of network’s (UDG) connectivity. Theorem 2 investigates the bounds of $|MDS|$, $|CDS|$, and $|MCDS|$ given that some parameters are known in advance. Proposition 1 can be used to predict the average size of an IS or CDS (constructed by mIS+Connection styled algorithms). Several constant factor algorithms have been proposed. Some of those algorithms are self functioned for special tasks, and together, they can provide solutions for CDS problem in UDGs. Simulation results show that our algorithms perform better than some known schemes. We show that approximation factors do not reflect the actual performance of the CDS constructed by mIS+Connection styled algorithms, and some currently best known factor possessing algorithms perform poor in practice. The reasons for such facts have also been discussed. Finally, to our knowledge, this is the first time to design approximation algorithms for CDS problem using geometric features as heuristics.

The geometry approach is an interesting research direction which deserves more attentions to be paid for. Our future works include: how to make an efficient use of the geometric features; how to improve the efficiency of the implementation, especially for distributed cases; and some theoretical issues, for example the relationship between time/message complexity (or approximation factors) and the size of CDS in UDGs, should be addressed; etc.

Finally, based on the observations on the extensive simulation results, we conjecture that approximation factor of CDS in UDG constructed by a centralized greedy algorithm like greedy area or Guha’s algorithm is no more than 2.

References

Gang Lu was born in Shanxi Province, China, in 1972. He received M.S. degree from Northwestern Polytechnical University, Xi’an, in 1999. He is currently pursuing the Ph.D. degree with School of Computer Science and Engineering in University of Electronic Science and Technology of China (UESTC). His research interests include computational geometry and graph theory.

Ming-Tian Zhou was born in Guangxi Province, China, in 1939. He is a professor and doctoral supervisor with the School of Computer Science and Engineering, UESTC. His current research interests include network computing, information security, and wireless networks.

Yong Tang was born in Yunnan Province, China, in 1973. He received Ph.D. degree in 2007 from School of Computer Science and Engineering, UESTC. He is now a lecturer with UESTC. His current research interests include network computing and wireless networks.

Ming-Yuan Zhao was born in Sichuan Province, China, in 1940. He is currently pursuing the Ph.D. degree with School of Computer Science and Engineering, UESTC. His current research interests include artificial immune system, and computational intelligence.

Xin-Zheng Niu was born in Guizhou Province, China, in 1978. He received Ph.D. degree in 2008 from School of Computer Science and Engineering, UESTC. He is now a lecturer with UESTC. His current research interests include mobile P2P networks and security.

Kun She was born in Sichuan Province, China, in 1967. He is a professor and doctoral supervisor with the School of Computer Science and Engineering, UESTC. His current research interests include middleware techniques and information security.