On Practical Construction of Quality Fault-tolerant Virtual Backbone in Homogeneous Wireless Networks

Bei Liu, Wei Wang, Donghyun Kim, Senior Member, IEEE, Yingshu Li, Senior Member, IEEE, Sung-Sik Kwon, Yaolin Jiang

Abstract—Over years, many efforts are made for the problem of constructing quality fault-tolerant virtual backbones in wireless networks. In case that a wireless network consists of physically equivalent nodes, e.g. with the same communication range, unit disk graph (UDG) is widely used to abstract the wireless network and the problem is formulated as the minimum $k$-connected $m$-dominating set problem on the UDG. So far, most results are focused on designing a constant factor approximation algorithm for this NP-hard problem under two positive integers $k$ and $m$ satisfying $m \geq k \geq 1$ and $k \leq 3$. This paper introduces an approximation algorithm for the problem with $m \geq k \geq 1$. This algorithm is simple to implement, connect the components by adding a bounded number of paths, which first computes a 1-connected $m$-dominating set $D$ and repeatedly (i) search the separators arbitrarily in $(i-1,m)$-CDS with $i = 2, 3, \cdots, k$, (ii) add a bounded number of paths connecting the components separated by separators in $(i-1,m)$-CDS to improve the connectivity of $(i-1,m)$-CDS, until it becomes $k$-connected, and (iii) remove redundant paths if there exist at every iteration. We provide a rigorous theoretical analysis to prove that the proposed algorithm is correct and its approximation ratio is a constant, for any fixed $k$.

Index Terms—Wireless networks, approximation algorithm design, fault-tolerant, connected dominating set, virtual backbone.

I. INTRODUCTION

Wireless networks such as wireless ad-hoc networks and wireless sensor networks are composed of numerous wireless mobile nodes, and have a number of important applications such as environment and habitat monitoring, traffic control, health applications, etc. [14], [15]. In most cases, a wireless node is battery operated and thus has a limited power source. In wireless communication, the amount of energy consumed for a node to transmit a message to another node increases super-linearly proportional to the distance between them. As a result, most wireless networks prefer multi-hop communication over long range direct communication to conserve its energy.

Unfortunately, the multi-hop communication strategy increases the number of messages flying over the network drastically and causes a huge amount of wireless signal interference and collision. As a result, the nodes consume much of its energy for retransmitting messages and waste lots of energy. This problem is known as the broadcasting storm problem and is a serious but inheriting issue in multi-hop routing wireless networks [16]. To ease the impact of the problem, one promising strategy would be having a backbone-like structure in the wired counterpart so that the number of nodes which are involved in the routing can be reduced. Based on the observation, Ephremides et. al [17] suggested to establish a subset of wireless nodes to be in charge of routing messages while the other nodes are not. Nowadays, this subset is called as a virtual backbone (VB) of the wireless network. Recent studies show that in addition to improve the efficiency of the wireless network, virtual backbone is known to bring several advantages to wireless networks as its adoption can be used to alleviate routing overhead and serve as an efficient platform for unicast, multicast, and fault-tolerant routing.

A subset of nodes in the unit disk graph (UDG) representing a wireless network of interest can be a VB in the graph if (a) the subgraph of the UDG induced by the subset is connected and (b) all nodes are either in the subset or adjacent to a node in the VB. In theory, the subset of a graph satisfying the requirements is referred as the connected dominating set (CDS). Apparently, a CDS of a UDG is better than another CDS of the UDG if its size is smaller as that means the CDS will suffer less from wireless signal interference and collision. Thus, Guha and Khuller [18] modelled the problem of computing quality virtual backbone as the minimum connected dominating set (MCDS) problem. The (minimum) dominating set problem is a well-known NP-hard problem and a generalized version of the MCDS problem as it does not require the induced graph by the subset connected. As a result,
the MCDS problem is also NP-hard, which implies that it is impractical to compute an optimal solution of a given MCDS problem, i.e. a minimum size CDS. As a result, many efforts are made to design and analyze an approximation algorithm for the problem, which has a worst-case performance guarantee [1], [2], [3], [4], [5].

In many wireless networks such as wireless sensor networks, nodes are subject to fail due to many reasons such as battery exhaustion or hostile environment. Due to the reason, the virtual backbone for wireless network is desirable to have some degree of fault-tolerance, in particular against node failure. In theory, a graph is k-(vertex)-connected if the graph is still connected after the removal of any k - 1 nodes, and a CDS whose induced graph has a higher level of fault-tolerance against such node failure. In [23], Dai and Wu investigated the requirements for a fault-tolerant virtual backbone and introduced the concept of the k-connected k-dominating set, or in short, (k, k)-CDS for a given k for the first time in the literature. In the later discussions, this concept is generalized as the k-connected m-dominating set, (k, m)-CDS. Formally speaking, a node subset is a k-connected m-dominating set if the following requirements are satisfied: given a network graph $G = (V, E)$ such as UDG, where $V$ is the set of nodes and $E$ is the set of edges, (a) $G[D]$, the subgraph of $G$ induced by a node subset $D$, is k (vertex) connected, i.e. $G[D \setminus D']$ is still connected for any $D' \subseteq D$ such that $|D'| \leq k$, and (b) $D$ is a m-dominating set in $G$, i.e. for any $u \in V \setminus D$, $u$ has at least m neighbors in $D$.

The minimum (cardinality) (k, m)-CDS problem is NP-hard as its special case with $k = 1, m = 1$, which is the MCDS problem, is NP-hard. Over recent years, many efforts are made to design a constant factor approximation algorithm to construct highly fault-tolerant virtual backbone. Note that by definition, $m \geq k \geq 1$ is desirable, otherwise, the failures of m nodes will fail the virtual backbone (i.e. some nodes will lose all of its neighboring backbone nodes) even though the backbone is fully operational (i.e. the subgraph induced by the residual backbone nodes is still connected). Wang et al. [37] proposed a 2-approximation algorithm for computing the minimum (2, 1)-CDS problem; Shang et al. [35] introduced a constant factor approximation algorithm for the minimum (2, m)-CDS problem, where $m \geq 1$. At INFOCOM 2010, 2015, 2016, Kim et al. [11], Wang et al. [6], Zhang et al. [7] studied the problem of constructing a constant factor approximation algorithm for the minimum (3, m)-CDS problem, Wang et al. [10] studied the problem of constructing a constant factor approximation algorithm for the minimum (4, m)-CDS problem, where $m \geq 4$. However, the question of designing a constant factor approximation algorithm for the minimum (k, m)-CDS problem is challenging for the case with any $k$ and $m \geq k \geq 5$.

To address this issue, this paper proposes a new approximation algorithm for the minimum (k, m)-CDS problem with $m \geq k \geq 1$. Roughly speaking, the algorithm is a round based one and each round consists of the following three steps, where $i$ is initially 2 and grows up to $k$.

(a) compute a $C_{i-1,m}$, which is a $(i - 1, m)$-CDS of a given UDG,
(b) find the $(i - 1)$-vertex separator of $C_{i-1,m}$,
(c) add a bounded number of $H$-paths to connect these components, which is separated by $(i-1)$-vertex separator.
(d) remove redundant $H$-paths if exist.

As a result, we obtain a new simpler constant factor approximation algorithm for the minimum $(k, m)$-CDS problem given any $m, k$ pair such that $m \geq k \geq 1$.

**Remark 1.** When preparing a revision of this paper, we noticed that Fukunaga [12] designed an approximation algorithm for $(k, m)$-CDS problem for $m \geq k \geq 1$ which is similar in spirit as ours.

Table I gives a globe view of approximation algorithms for the minimum $(k, m)$-CDS problem for different cases. The reminder of this paper is organized as follows. Several important notations and definitions are provided in Section II. Section III introduces some related work. In Section IV, we present a new constant factor approximation algorithms for the minimum $(k, m)$-CDS problem in UDG and also give a theoretical analysis of its performance ratio. Finally, we conclude this paper and discuss some future research directions in Section V.

II. Notations and Definitions

This section introduces some important definitions and notations.

**Definition 1** (Unit Disk Graph (UDG)). A graph $G = (V, E) = (V(G), E(G))$ on a 2-dimensional euclidean space is referred as a UDG, if for any pair $u, v \in V$, there exists a bidirectional edge between them only if the euclidean distance between them is not greater than 1, i.e. $udist(u, v) \leq 1$.

**Definition 2** (k-Vertex-Connected Graph). In graph theory, a graph $G = (V, E)$ is said to be a k-vertex-connected graph (or k-connected) if it has more than k vertices and k is the size of the smallest subset of vertices such that the graph becomes disconnected if the subset is deleted. An equivalent definition is that a graph $G$ with at least two vertices is k-connected if, for every pair of its vertices, it is possible to find k node-disjoint paths connecting these nodes.

**Definition 3** (m-Dominating Set). Given a graph $G = (V, E)$, we call a vertex $v \in V$ dominated by a vertex $u \in V$ if there exists an edge $(u,v) \in E$. A subset $D \subseteq V$ is an m-dominating set if every vertex $v$ in $V \setminus D$ is dominated by at least m vertices in $D$.

**Definition 4** (k-Connected m-Dominating Set). A subset $D \subseteq V$ is a k-connected m-dominating set $(k, m)$-CDS of graph $G = (V, E)$ if, (i) $D$ is an m-dominating set of $G$ and (ii) the induced subgraph $G[D]$ is k-connected.

In this paper, $(k, m)$-CDS is abbreviated as $C_{k,m}$, and both represent a k-connected m-dominating set.

**Definition 5** (Separating Set or Vertex Separator or Vertex Cut). $G = (V, E)$ is a k-connected graph, if there exists a subset $S \subseteq V$, and $|S| = k$, whose removal renders the induced graph $G \setminus S$ disconnected, then $S$ is called a separating set (or vertex separator, vertex cut).
According to the definition of a $k$-connected graph, we know that, in a $k$-connected graph, the vertex separator whose size is more than $k$ can also disconnect the graph. For the sake of the simplicity of our later discussion, we have the following definition:

**Definition 6 (i-Vertex Separator).** $G = (V, E)$ is a $k$-connected graph, if there exists a subset $S = \{s_1, s_2, ..., s_i\} \subseteq V$ where $i \geq k$, and then $G \setminus S$ is disconnected, $S$ is called an i-vertex separator.

In this paper, for better understanding, in a $k$-connected graph, both separating set and separator are used to represent $k$-Vertex Separator.

**Definition 7 (Cross Separating Set).** $G = (V, E)$ is a $k$-connected graph, $S \subset V$ is a separating set, which separates $G$ into $C_1, C_2, ..., C_k$. If $S = \{s_1, s_2, ..., s_k\}$ is the separating set of $G$ and $S' \not\subset C_i$ for any $i = 1, 2, ..., k$, then we call $S'$ to be the separating set. In other words, $S'$ consists of the vertices from different components separated by $S$. $S'$ is called cross separating set regarding $S$.

Fig. 1 is an illustration of cross separating set.

**Definition 8 ((a, b)-Separator).** In graph theory, given a graph $G = (V, E)$, a vertex subset $S \subset V$ is a vertex cut (or vertex separator, separating set) for two non-adjacent vertices a and b if the removal of S from the graph disconnects a and b. Then we call S an (a, b)-separator.

In Fig. 1, $\{v_1, v_9, v_5\}$ is a $(v_i, v_j)$-separator where $i \in 2, 3, 4$ and $j \in 6, 7, 8$; $\{v_3, v_7, v_9\}$ is $(v_i, v_j)$-separator where $i \in 4, 5, 6$ and $j = 1, 2, 8$.

**Definition 9 (H-path).** Given a graph $G = (V, E)$, an H-path $P$ of a subgraph $H$ of $G$ is a path between two different nodes in $H$ such that all of the inner nodes of $P$ are not in $H$.

**Definition 10 (The Augmentation Problem).** Given a $k$-connected graph $G = (V, E)$, and a $(k - 1, m)$-CDS $D$ of $G$, find a subset $H \subset V \setminus D$ with minimum size such that the induced subgraph $G[D \cup H]$ is $k$-connected.

### III. Related work

Over years, the concept of dominating set in a graph attracted lots of attentions and even recently, a number of its applications are actively investigated in the literature [46], [47], [48], [49], [50], [51]. During the past years, a lot of effort has been made to design approximation algorithms for fault-tolerant connected dominating set problems. Several approximation algorithms for constructing $(k, m)$-CDS have been proposed in the literature. The problem of constructing fault-tolerant virtual backbone was first proposed by Dai and Wu [23]. They proposed three heuristic algorithms colorgreeno compute $k$-connected CDS construction. However, none of them guarantees the size bound of the resulting CDSs.

In [25], Alzoubi et al. proposed an approximation algorithm to construct a minimum CDS with performance ratio of 8. In [28], Li et al. provided a distributed algorithm for computing $r$-CDS whose performance ratio is 147. Wang et. al. introduced Connecting Dominating Set Augmentation (CDSA) in [37]. They proposed a 64-approximation centralized algorithm to construct a 2-connected dominating set. This algorithm first constructs a CDS, and then computes all the blocks and adds intermediate nodes to make the backbone 2-connected. Further, in [35], Shang et al. proposed a centralized algorithm to construct 2-connected $k$-dominating set. They first choose an MIS of $G$, then choose an MIS $k$ times, a $k$-dominating set $G[D]$ can be obtained, finally add $H$-paths to make $G[D]$ 2-connected. This algorithm has performance ratio depending on $k$. Recently, Shi et al. proposed a greedy algorithm for computing minimum 2-connected $m$-dominating set [36], which has a performance ratio of 12.89 for $m \geq 5$.

Thai et. al. first introduced a centralized approximation algorithm to compute $k$-connected $m$-dominating set [34]. The main idea is: first compute a 1-connected $m$-dominating set; then calculate a $k$-connected $k$-dominating set based on the first step; at last, construct a $k$-connected $m$-dominating set. In [40], Wu et. al. proposed a centralized approximation algorithm, CGA, and a distributed algorithm, DDA, to construct $k$-connected $m$-dominating sets for any $k$ and $m$. These two algorithms CGA, DDA are improved by the algorithms CGA and DDA, respectively in [41].

Recently, in [11], [42], the authors proposed a new polynomial time algorithm for computing $(3, m)$-CDS. They first, compute a $(2, 3)$-CDS, then iteratively convert the bad point to a good point by adding $H$-paths. The performance ratio of this algorithm is 280. This result is improved greatly by work [6], the first algorithm apply Tutte decomposition to design a basic algorithm, which first decomposing the $(2, m)$-CDS into bricks and add $H$-paths to construct the $(3, m)$-CDS; The second one is simpler to implement. The performance ratios of these algorithms are 87 and 62, respectively. In [7], Zhang et.al used the greedy strategy to design an algorithm for computing $(3, m)$-CDS, whose performance ratio is 26.34.
In graph theory, a block is a maximal component (i.e., connected subgraph) without separating set (a subset of nodes without which, the residual graph becomes disconnected). A block-tree of a graph is induced by the graph by contradicting every block into a node, and is a very useful tool to improve the connectivity of a \((k, m)\)-CDS. For example, in [37], cut-vertices decompose a CDS into leaf-blocks and blocks to augment a \((1, 1)\)-CDS to a \((2, 1)\)-CDS. In [6], the unique Tutte decomposition of 2-connected graphs used to augment \((2, m)\)-CDS to \((3, m)\)-CDS. As Tutte decomposition is only applicable to 2-connected graph. This strategy fails to construct \((k, m)\)-CDS for \(k > 3\).

Very recently, Wang et. al [10] introduced a new constant factor approximation algorithm for the problem of computing the minimum \((4, m)\)-CDS. A new strategy is introduced. First, \((3, m)\)-CDS is computed. Then the algorithm uses auxiliary graphs with simpler structure to verify the vertex cuts of the complicated original graphs. Further, the vertex cut can be identified by searching degree-\((k - 1)\) point. Then by adding \(H\)-paths, the connectivity of the graph is increased. However, the auxiliary graphs are simplified by removing the removable point, which requires the existence of the removable point. So, this strategy cannot be used for computing \((k, m)\)-CDS for \(k > 4\).

The work most related to ours is Fukunaga [12], in which the author gave, among others, a constant approximation to \((k, m)\)-CDS for any \(m \geq k \geq 1\). The algorithm is based on the same idea as [10], while the augmentation of the connectivity step is slightly different, i.e., at every time, a minimal demand cut (see IV B for details) is searched and found, then an \(H\)-path is added to remove the corresponding separator, until no separator left. We mention in passing that independent of Fukunaga [12], most recently, Shi et al. [8] gave a constant approximation algorithm for \((k, m)\)-CDS for any \(m \geq k \geq 1\) in a journal version of their conference paper [7], which is totally different from ours.

However, we noticed that instead of using a minimal demand cut to find the \(H\)-paths added, we are able to show that finding an arbitrary separator at each time suffices to guarantee that a bounded number of \(H\)-paths is added at each iteration, so that the resulting algorithm has a constant approximation ratio (which is the same as that in [12]). This was achieved by showing that the output of our algorithm can always be realized by that in [12], by using a clever and non-trivial argument. Accordingly, our algorithm is much simpler to be implemented than that in [12].

Table I gives a summary of the existing works and presents a comparison of performance ratios in references and Algorithm 1 with specific \(k\). In Table I, the main idea in [35] [11] [6] is the block tree, the strategy used in [36] and [7] is greedy algorithm, in [10], the auxiliary graph is used to solve this problem.

IV. MAIN RESULTS

In this section, we propose our constant factor approximation algorithm for the \(k\)-connected \(m\)-dominating set problem with \(m \geq k \geq 1\).

A. A new constant factor approximation algorithm for the minimum \((k, m)\)-CDS

In this section, we present our new constant factor approximation algorithm for computing \((k, m)\)-CDS. Our main idea is adding \(H\)-paths to remove \((i - 1)\)- separators successively until there is no separator left. The rough outline of this algorithm is as follows:

(a) Given an input UDG \(G = (V, E)\) which has a feasible solution (and therefore it is \(k\)-connected), the algorithm first computes a \((1, m)\)-CDS \(D\) using an existing algorithm.

(b) For each \(i = 2\) to \(k\), we repeat the following steps until \(G[D]\) becomes a \((k, m)\)-CDS:

(i) Let \(D_0 \leftarrow D\) (Notice that \(D_0\) will be fixed throughout the \(i\)-th round). Recursively search and find the \((i - 1)\)-vertex separator in \(G[D_0]\), which decompose \(G[D_0]\) into a set of components (connected subgraphs). In detail, initially, check if \(G[D_0]\) has an \((i - 1)\)-vertex separator, and if so, \(D_0\) is split into three subsets: \((i - 1)\)-vertex separator \(S\), and two nodes subsets \(D_L\) and \(D_R\) which are separated by \(S\), i.e. \(G[D_L]\) and \(G[D_R]\) are disconnected.

(ii) Add an \(H\)-path (a path from a node \(u\) in \(D_L\) to another node \(v\) in \(D_R\)) with length at most 3 to \(G[D_0]\) so that the \((i - 1)\)-vertex separator \(S\) can be removed.

(iii) Repeatedly search and find if there is an \((i - 1)\)-vertex separator in \(G[D_0]\). And continue to repeat Step (bii) until there is no \((i - 1)\)-vertex separator \(S\) in \(G[D_0]\).

(iv) Let \(H = \{h_1, h_2, \ldots, h_l\}\) denote the set of all the \(H\)-paths added, let \(D \leftarrow D_0 \cup H\), where \(H\) denotes the set of all the inner nodes of the \(H\)-paths in \(H\). If \(G[(D \cup H) \setminus h_j]\) is \(i\)-connected, then remove \(h_j\). (In this step, the redundant \(H\)-paths are removed from \(H\). Thus, after this step, we can obtain a minimal \(i\)-connected subgraph.)

(v) As a result, the connectivity of \(G[D]\) is increased by 1 (from \(i - 1\) to \(i\)).

After the execution of the algorithm, a \((k, m)\)-CDS is returned.

B. Performance Analysis

In this section, we provide a theoretical analysis to show Algorithm 1 is in fact a constant factor approximation algorithm for computing the minimum \((k, m)\)-CDS.

As presented, in our Algorithm 1, once a separating set is identified, an \(H\)-path is added. In a \(k\)-connected graph \(G = (V, E)\) on \(n\) vertices. Clearly, in the worst case, any \(\{s_1, s_2, \ldots, s_k\} \subset V\) may consist of a separating set. Thus, the number of \(k\)-vertex separator sets in a graph \(G\) has a trivial upper bound of \(\binom{n}{k}\). This bound for any integer \(k\) can be achieved when there are \(n\) vertices without any edges in graph \(G\). In [44], the authors give an upper bound of the number of separating sets of \(k\)-connected graphs. For \(k = 2\), in an undirected biconnected graph with vertex size \(n\), the maximum number of separating sets is \(n(n-3)\), which is achieved by a cycle with \(n\) vertices. For \(k = 3\), the upper bound is
Algorithm 1 A New Constant Algorithm for Computing \((k, m)\)-CDS, \(m \geq k \geq 1\)

1: Input: a \(k\)-connected graph \(G = (V, E)\), and two integers \(k\) and \(m\).
2: Output: a \(k\)-connected \(m\)-dominating set \(D\).
3: Compute \((1, m)\)-CDS and set \(D = D_0 \leftarrow (1, m)\)-CDS.
4: for \(i = 2\) to \(k\) do
5: \quad if there is a \((i - 1)\)-Vertex Separator \(S\) in \(G[D_0]\) then
6: \quad \quad \text{repeat}
7: \quad \quad \quad Add \(H\)-paths to connect the components in \(G[D_0 \setminus S]\), \(H\) denotes the set of all the \(H\)-paths added.
8: \quad \quad \quad until there is no \((i - 1)\)-Vertex Separator in \(G[D_0]\).
9: \quad end if
10: \quad \quad \text{end for} \(D = D \cup H\), where \(H\) is the set of all inner nodes of \(H\)-paths in \(H = \{h_1, h_2, ..., h_t\}\).
11: \quad for \(j = 1\) to \(l\) do
12: \quad \quad if \(G[D \cup H \setminus h_j]\) is \(i\)-connected, then
13: \quad \quad \quad \text{repeat}
14: \quad \quad \quad \quad Delete \(h_j\) from \(H\).
15: \quad \quad \quad \quad until there is no redundant \(H\)-path
16: \quad \quad end if
17: \quad end for
18: \(D_0 \leftarrow D\).
19: end for
20: Return \(D\)

\((n-1)(n-4)/2\), and wheel graph achieves it. For general \(k\), the upper bound is \(O(2k/3n^2)\).

The performance of our algorithm depends on how many \(H\)-paths are added in order to increase the connectivity. In the worst case, each separating set requires one \(H\)-path. In Algorithm 1, our initial goal is to remove some specific separating set by adding an \(H\)-path. However, after this \(H\)-path has been added, it may bring additional benefit: other separating sets are removed simultaneously. In [9], when we aim to augment the connectivity of a biconnected graph, we remove the separators one by one by adding \(H\)-paths. With the process of removing specific separators, other separators will be removed at the same time. Therefore, when the biconnected graph becomes triconnected, the number of \(H\)-paths added is far less than that of the separators in the original graph.

As a result, with the influence of the \(H\)-paths, the number of \(H\)-paths needed to remove all the separators can be reduced from \(O(n^2)\) to \(O(n)\). This explains intuitively why our strategy works.

In order to give a theoretical analysis of the performance ratio of Algorithm 1, next, we discuss the upper bound of the number of \(H\)-paths required to remove all \((i - 1)\)-vertex separators in an \(i\)-connected \(m\)-dominating set at the \(i\)th round. To this ends, we need to introduce several definitions below.

Given a \(k\)-connected graph \(G = (V, E)\), for \(X \subset V\), \(\Gamma(X)\) denotes the set of neighbors of \(X\) in \(D\). In other words, \(\Gamma(X) = \{v \in V \setminus X : uv \in E\text{ for some } u \in X\}\). \(X^+\) denotes \(X \cup \Gamma(X)\) for any \(X \subset V\). For \(X, T \subset V\), \(\Gamma_T(X) = \Gamma(X) \cap T\).

**Definition 11** (T-cut). Let \(X \subset V\). For \(T \subset V\), \(G[T]\) is \((k - 1)\)-connected. \(X\) is called a T-cut if \(X \subset T\) and \(X \neq \emptyset \neq T \setminus X^+\).

**Definition 12** (Demand cut). A T-cut \(X\) is called a demand cut, if \(|\Gamma_T(X)| = k - 1\). Furthermore, \(X\) is a minimal demand cut if there is no demand cut \(Y \subset V\) with \(Y \subset X\).

**Definition 13** (an H-path h covers a separator S). Let \(D_0\) be an \((i - 1, m)\)-CDS and \(G[D_0]\) is not \(i\)-connected. Let \(S\) be a separator of \(G[D_0]\). Suppose that \(G[D_0 \setminus S]\) is not \(i\)-connected. If \(h\) is an \(H\)-path such that the two endpoints joint two different connected components of \(G[D_0 \setminus S]\). Then we say the \(H\)-path \(h\) covers the separator \(S\).

**Remark 2.** In the above definition, if \(r > 2\), we duplicate \(S\)
with \( r - 1 \) copies of \( S \), and regard \( S \) having multiplicity \( r - 1 \). In this case, \( r - 1 \) \( H \)-paths of length at most three are needed to completely removed the separator \( S \).

Let \( S_h \) denote all the separators that can be covered by the \( H \)-path \( h \), let \( H_S \) denote the set of the \( H \)-paths which can cover the separator \( S \). In some cases, if all the separators covered by the \( H \)-path \( h_1 \) can also be covered by the \( H \)-path \( h_2 \) and \( h_2 \) can cover some separators not in \( S_{h_1} \), i.e., \( S_{h_1} \subset S_{h_2} \). We say that the \( H \)-path \( h_2 \) covers the \( H \)-path \( h_1 \).

Now, we briefly introduce the algorithm in [12], and we will show this algorithm is equivalent to our algorithm. The main steps of the algorithm \( B \) in [12] are as follows:

(a) Given an input \( k \)-connected \( G = (V, E) \), and a \((k-1,m)\)-CDS \( T \subseteq V \).

(b) If there is a minimal demand cut \( X \) with \( \Gamma(X) = k-1 \), Add \( H \)-paths (a path from a node \( u \) in \( X \) to another node \( v \) in \( T \setminus X^+ \)) with length at most 3 to \( G[T] \). Add the inner nodes in the \( H \)-paths in \( T \).

(c) Repeatedly research and find if there is a minimal demand cut \( X \) in \( G[T] \). And continue to repeat Step (b) until there is no demand cut.

Different from Algorithm \( B \), our algorithm only needs to find the separator directly rather than to find the minimal demand cut. Actually, given a graph \( G = (V, E) \), and \((i-1,m)\)-CDS \( D \), let \( D_0 \leftarrow D \). We can find all the \((i-1)\)-vertex separators in \( D_0 \). Let \( S_1, S_2, \ldots, S_p \) denote all the \((i-1)\)-vertex separators of \( G[D_0] \) and let \( S = \{S_1, S_2, \ldots, S_p\} \). Further, we can find all the \( H \)-paths \( h_1, h_2, \ldots, h_{r_S} \), which can cover \( S \).

In the same way, we can find the \( H \)-paths \( h_1, h_2, \ldots, h_{r_S} \), which can cover \( S_i \), for \( i = 2, \ldots, p \). Let \( H = \{h_1, h_2, \ldots, h_q\} \) denote the set of all the \( H \)-paths, which can cover some separators in \( S \), i.e., \( H = \{h_1^1, h_2^1, \ldots, h_{r_S}^1, h_1^2, h_2^2, \ldots, h_{r_S}^2, \ldots, h_1^p, h_2^p, \ldots, h_{r_S}^p\} \).

It should be noticed that one \( H \)-path may cover several separators, and one separator can be covered by several \( H \)-paths. For the ease of presentation, we can construct a new graph \( G' = (V', E') \) as follows:

1. Let \( V' = S \cup H \).
2. If an \( H \)-path \( h_i \) can cover a separator \( S_j \), add an edge between \( h_i \) and \( S_j \).

By the construction, the graph \( G' = (V', E') \) is a bipartite graph. Moreover, this graph provides a global view regarding the relationship between the separators and the \( H \)-paths; see Fig 3.

From the above discussions, we know that, in our algorithm, once an \((i-1)\)-vertex separator \( S_1 \) was found, the algorithm will select one \( H \)-path, say \( h_1 \), to cover \( S_1 \). If there is an additional separator in \( S_{h_1} \), then after adding \( h_1 \), several \((i-1)\)-vertex separators are covered (removed). Then we find the next \((i-1)\)-vertex separator in \( S \setminus \{S_1\} \), where \( S = \{S_1, S_2, \ldots, S_p\} \). Thus, the process of finding separators and adding \( H \)-paths to \( G[D_0] \) in Algorithm 1 at the \( i \)-th round can be modeled as the Set MultiCover Problem, in which the ground set is \( S \), an element (a separator) in \( S \) has to be covered a required times (the multiplicities of the separator).

The advantage of this global view is apparent: it provides a whole picture of adding \( H \)-paths and removing separators in the algorithm; see Fig. 6 for illustrations, where the star denotes a separator, the cycle denote an \( H \)-path, and the stars inside the circle are covered by the \( H \)-path.

However, for the same input graph, Algorithm 1 may return different solutions. Because when one separator was found, we have several \( H \)-paths as options, and the option will affect the choice of the next separator selected. The different solutions of Algorithm 1 correspond to different set cover; see Fig. 4 and Fig. 5 for illustrations.

From the above, we know that each solution of Algorithm 1 can be represented by a Set Cover. And from our algorithm,
each H-path which is added must cover at least one \((i - 1)\)-vertex separator which cannot be covered by all the other H-paths added. In other words, if we delete any H-path added, the output graph is not \(i\)-connected (since no redundant H-path exists).

If Algorithm 1 outputs a solution with the \((i - 1)\)-vertex separators \(S_1, S_2, \ldots, S_l\) and the H-path \(h_1, h_2, \ldots, h_l\), Algorithm B will return the same solution by using the following strategy:

(a) First, select a minimal demand cut \(X_1\) with \(|\Gamma(X_1)| = i - 1\). Then \(\Gamma(X_1)\) is a separator in \(S\) which can be covered by some H-paths \(t_1 \in H = \{h_1, h_2, \ldots, h_l\}\), since the H-paths in \(S\) covers all the separators including \(\Gamma(X_1)\).

(b) Then, select the second minimal demand cut \(X_2\) with \(|\Gamma(X_2)| = i - 1\) such that \(\Gamma(X_2) \notin S_{t_1}\), i.e., \(t_1\) cannot cover \(\Gamma(X_2)\). Then there must exist an H-path \(t_2 \in \{h_1, \ldots, h_l\} \setminus \{t_1\}\) that covers \(\Gamma(X_2)\).

(c) Repeatedly select the minimal demand cut, say, \(X_1, X_2, \ldots, X_l\) are the ones obtained. Then H-paths \(t_1, \ldots, t_l\) cover \(\Gamma(X_1), \Gamma(X_2), \ldots, \Gamma(X_l)\).

From the strategy, we have the following

**Lemma 1.** For any solution of Algorithm 1, the same solution can be output by Algorithm B.

**Proof.** Suppose upon running Algorithm 1, we get a solution with the \((i - 1)\)-vertex separators \(S_1, S_2, \ldots, S_l\) and the H-paths \(h_1, h_2, \ldots, h_l\), and let \(S = \{S_1, S_2, \ldots, S_l\}\) and \(H = \{h_1, h_2, \ldots, h_l\}\), where \(S_h\) denotes the \((i - 1)\)-vertex separators that can be covered by \(h_i\). Firstly, according to the above strategy, the demand cut \(X_i\) \((i = 1, \ldots, l)\) is the minimal one in each step, and then, in order to guarantee the demand cut \(X_i\) to be minimal, the corresponding separator \(\Gamma(X_i)\) cannot be the \(i\)-th \((i - 1)\)-vertex separators selected. But we always can select the H-path \(t_i = h_j\) to cover the \(\Gamma(X_i)\). In the last step of Algorithm 1, we delete the redundant H-paths, so the H-path that added first will not increase the number of the H-paths added. Thus, Algorithm B uses the set of H-paths \(H = \{t_1, t_2, \ldots, t_l\}\) to cover all the separators. Therefore, Algorithm 1 and Algorithm B use same set of H-paths to cover all the \((i - 1)\)-vertex separators.

By Lemma 1 and from the previous discussion, we have the following lemma.

**Lemma 2.** The number of the H-paths added in Algorithm 1 does not exceed the number of the H-paths added in Algorithm B.

**Proof.** Because every solution obtained by Algorithm 1 is corresponding to a solution of Algorithm B with the same H-paths.

Fig. 3 - Fig. 9 give an illustration of the strategy. Fig 6 is the solution of the Algorithm 1, Fig 9 presents the solution of Algorithm B. From these two figures, we can know that, although the separators found in Algorithm 1 are different from the separators in Algorithm B, by our method, we can obtain the same solution output by Algorithm 1 and Algorithm B. Refer to Fig 6 and Fig 9, different red stars are selected, but the same H-paths are added.

**Lemma 3 ([12]).** For \(i = 3, \ldots, k\), the number of the H-paths added in \(G[D_0]\) can be bounded by \(i(2|D_0| - 3)\).

**Proof.** In [12], we know there are at most \(i(2|D_0| - 3)\) minimal demand cut. Thus, at most \(i(2|D_0| - 3)\) H-paths needed to add to increase the connectivity.
Lemma 4. Algorithm 1 is a 3r-approximation for computing \((2, m)\)-CDS, where \(r\) is the approximation ratio for the minimum \((1, m)\)-CDS.

Proof. There are \((n - 2)\) vertex cuts at most in a connected graph \(G = (V, E)\) with the node size of \(n\) [44]. Thus, in Algorithm 1, only need to add \(|D|\) \(H\)-paths to make \(D\) 2-connected.

Lemma 5. For any \((i - 1)\)-connected \(m\)-dominating set \(D\) of graph \(G = (V, E)\), let \(x\) be a new vertex which is added in \(D\) and adjacent to at least \(i\) vertices in \(D\), then addition of \(x\) will not introduce any new separating set.

Since the node added into \(C_{i-1,m}\) is at least \(m\)-dominated by \(C_{i-1,m}\), we have the following

Lemma 6. The number of the intermediate nodes of the \(H\)-path is at most two.

In Algorithm 1, in Line 5 - Line 8, once there exists an \((i - 1)\)-vertex separator, add \(H\)-paths to remove it until there is no \((i - 1)\)-vertex separator. Thus, for any \(i = 2, \ldots, k\), the connectivity of \(D\) improves until the connectivity is \(k\). Thus, we have the following theorem:

Theorem 1. The output \(D\) is \(k\)-connected \(m\)-dominating set of \(G = (V, E)\).

Theorem 2. Algorithm 1 is a constant factor approximation for computing \((k, m)\)-CDS.

Proof. For \(i = 2, \ldots, k\), after the \((i - 1, m)\)-CDS \(D_0\) is constructed, from Lemma 3, Lemma 4, there are at most \(2|D_0|\) \(H\)-paths added to make \(D_0\) to be \(i\)-connected. Thus, from Lemma 6, there are at most \(4|D_0|\) nodes added.

\(C_{i,m}\) denotes the optimal solution for \(i\)-connected \(m\)-dominating set. \(H_1\) denotes the set of all the nodes added when there is no separating set in \(C_{i-1,m}\). Thus,

\[
|C_{i,m}| \leq |C_{i-1,m}| + |H_1| \leq n + 4in \leq 5in
\]

As \(i\) varies from 2 to \(k\), each augmentation incurs approximation factor of is \(3r \times 5^{k-2}k!\) with \(k \geq 3\), where \(r\) is the approximation ratio for the minimum \((1, m)\)-CDS.

Theorem 3. Let \(n\) be the number of nodes in the original graph \(G = (V, E)\). Then the time complexity of Algorithm 1 is \(O(n^3)\).

Proof. Since the sequential implementation of the algorithm in [44] for verifying whether \(\{u, v\}\) is a separator runs in \(O(2^kn^3)\). And the time complexity of adding \(H\)-path is dominated by the Shortest-Path function, which runs in \(O(n^2)\). Therefore, the time complexity is at most \(O(2^kn^5)\).

V. CONCLUSION

In this paper, we investigated the problem of constructing highly fault-tolerant virtual backbone, i.e., computing a \((k, m)\)-CDS in wireless networks, where \(k\) and \(m\) are a pair of integers satisfying \(m \geq k \geq 1\). We have proposed a constant factor polynomial time approximation algorithm to compute \((k, m)\)-CDSs. In the future, we will focus on developing approximation algorithm for the minimum \(k\)-connected \(m\)-dominating set problem with better performance ratio and simpler structure by other strategy than augmentation.

ACKNOWLEDGEMENT

This research was jointly supported by National Natural Science Foundation of China under grants 11471005. This work was also supported by US National Science Foundation (NSF) No. HRD-1345219.

Bei Liu received the MS degree in mathematics from Xi’an Jiaotong University, Xi’an, China. Currently, she is pursuing the PhD degree in mathematics in Xi’an Jiaotong University. Her major research interests include wireless networking, social networking, and approximation algorithm design and analysis.
Wei Wang received the BS degree in applied mathematics from Zhejiang University, Hangzhou, China (1991). He received the MS degree in computational mathematics (1994) and PhD degree in mathematics from Xian Jiaotong University, Xian, China (2006). He is currently a professor at School of Mathematics and Statistics, Xian Jiaotong University. His research interests include algebraic graph theory and approximation algorithm design and analysis.

Donghyun Kim received the BS degree in electronic and computer engineering from the Hanyang University, Ansan, Korea (2003), and the MS degree in computer science and engineering from Hanyang University, Korea (2005). He received the PhD degree in computer science from the University of Texas at Dallas, Richardson, USA (2010). Currently, he is an assistant professor in the Department of Computer Science at Kennesaw State University, Marietta, USA. From 2010 to 2016, he was an assistant professor in the Department of Mathematics and Physics at North Carolina Central University, Durham, USA. His research interests include cyber physical, next generation computing and networking, and algorithm design and analysis. He is a member of ACM and a senior member of IEEE.

Yingshu Li received her Ph.D. and M.S. degrees from the Department of Computer Science and Engineering at University of Minnesota-Twin Cities. She received her B.S. degree from the Department of Computer Science and Engineering at Beijing Institute of Technology, China. Dr. Li is currently an Associate Professor in the Department of Computer Science at Georgia State University. Her research interests include Wireless Networking, Sensor Networks, Sensory Data Management, Social Networks, and Optimization. Dr. Li is the recipient of an NSF CAREER Award. She is a senior member of the IEEE.

Sung-Sik Kwon received the PhD degree in Mathematics from University of North Carolina at Chapel Hill in 1996. He is currently an associate professor in the Department of Mathematics and Computer Science at North Carolina Central University. His research interests include Numerical PDEs and nonlinear optimization.

Yaolin Jiang is a full Professor in the Department of Mathematics at Xian Jiaotong University since 1998 and now is also a Changjiang Professor in China. He has published 4 books and about 230 academic papers in journals. His research interests include theoretical studies of scientific computing, model order reduction, waveform relaxation, numerical solutions of partial differential equations, domain decomposition methods, matrices and tensors, dynamics of non-linear systems, circuit simulation and parallel processing.

References


