On Approximating Minimum 3-Connected m-Dominating Set Problem in Unit Disk Graph

Bei Liu, Wei Wang, Donghyun Kim, *Senior Member, IEEE, Member, ACM*, Deying Li, Jingyi Wang, Alade O. Tokuta, *Life Member, IEEE*, and Yaolin Jiang

***Abstract*—Over years, virtual backbone has attracted lots of attention as a promising approach to deal with the broadcasting storm problem in wireless networks. Frequently, the problem of a quality virtual backbone is formulated as a variation of the minimum connected dominating set problem. However, a virtual backbone computed in this way is not resilient against topology change since the induced graph by the connected dominating set is one-vertex-connected. As a result, the minimum k-connected m-dominating set problem is introduced to construct a fault-tolerant virtual backbone. Currently, the best known approximation algorithm for the problem in unit disk graph by Wang *et al.* assumes $k \leq 3$ and $m \geq 1$, and its performance ratio is 280 when $k = m = 3$. In this paper, we use a classical result from graph theory, Tutte decomposition, to design a new approximation algorithm for the problem in unit disk graph for $k \leq 3$ and $m \geq 1$. In particular, the algorithm features with (a) a drastically simple structure and (b) a much smaller performance ratio, which is nearly 62 when $k = m = 3$. We also conduct simulation to evaluate the performance of our algorithm.**

***Index Terms*—3-connected m-dominating set, approximation algorithm, fault-tolerant, Tutte decomposition, virtual backbone, wireless networks.**

I. Introduction

IT IS well known that energy efficiency is one of the most significant efficiency issue of wireless networks such as ad hoc networks and wireless sensor networks [2]. In many cases, each battery-operated wireless node consumes most of its energy for wireless communications. The study on the broadcast storm problem in wireless networks [3], which happens when a flooding-based routing algorithm is initiated to identify a routing path from a source to a destination, has demonstrated the importance of a sophisticated communication coordination to improve the energy efficiency of wireless networks. Recently, the idea of virtual backbone, which is a subset of nodes in a wireless networks, is in charge of managing routing information, and thus contributes to reduce routing-related overhead, wireless signal collision, and interference, and has emerged as a prominent approach to address this issue.

It is known that as the size of a virtual backbone decreases, it is getting more efficient. As a result, lots of attention has been paid to compute a virtual backbone with smaller size [4]–[9], [11], [27]. In theory, given a graph representing a wireless network, the problem of computing the minimum size virtual backbone can be abstracted as the minimum *connected dominating set* (CDS) problem. Since this problem is NP-hard, a significant amount of effort is made to design a polynomial-time heuristic algorithm with theoretical worst-case performance guarantee, which is also known as an approximation algorithm, for it.

In graph theory, a graph G is k-connected (more precisely, is k-vertex-connected) if G is still connected after removing any $k-1$ nodes from it, and by definition (see Definition 2 for details), a CDS is only guaranteed to be 1-connected. However, wireless networks have various applications in which their topologies are not stable due to many reasons such as uneven energy depletion at more active nodes or failures of nodes around hostile/hazardous environments. In [12], Dai and Wu identified that the virtual backbones for wireless networks with dynamic topology should have higher degree of fault tolerance, and introduced the concept of k-connected k-dominating set, where k is a positive integer given by the wireless network operator and represents the degree of desired fault tolerance (see Definition 5 for the formal definition).

Clearly, the problem of computing a minimum size k-connected k-dominating set is NP-hard since its special case with $k = 1$ is equivalent to the minimum CDS problem. Due to the reasons, lots of efforts are made to design approximation algorithms for the minimum k-connected m-dominating set problem, where k and m are two independent positive integers. In [16], Wang *et al.* introduced the first constant factor approximation algorithm for the minimum 2-connected 1-dominating set problem. In [15], Shang *et al.* proposed the first constant factor approximation algorithm for the minimum 2-connected m-dominating set problem for any positive integer

Manuscript received February 01, 2015; revised July 17, 2015; accepted August 26, 2015; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor P.-J. Wan. Date of publication September 22, 2015; date of current version October 13, 2016. This work was supported by the National Natural Science Foundation of China under Grants No. 11471005 and No. 11371287, in part by the US National Science Foundation (NSF) under Grants No. HRD-1345219 and No. HRD-1533653, and jointly supported by the Fundamental Research Funds for the Central Universities and the Research Funds of Renmin University of China 10XNJ032. The preliminary version of this paper appeared in the Proceedings of the 34th IEEE International Conference on Computer Communications (INFOCOM) 2015. *(Corresponding author: Donghyun Kim.)*

B. Liu, W. Wang, and J. Wang are with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: liubei@stu.xjtu.edu.cn; wang_weiw@163.com; 512467239@qq.com).

D. Kim and A. O. Tokuta are with the Department of Mathematics and Physics, North Carolina Central University, Durham, NC 27707 USA (e-mail: donghyun.kim@nccu.edu; atokuta@nccu.edu).

D. Li is with the Key Laboratory of Data Engineering and Knowledge Engineering, MOE, School of Information, Renmin University of China, Beijing 100872, China (e-mail: deyingli@ruc.edu.cn).

Y. Jiang is with the College of Mathematics and System Sciences, Xinjiang University, Urumqi 830046, China and with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: yljiang@mail.xjtu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2475335

m. Most notably, in [13] and [14], the authors introduced the first constant factor approximation algorithm for the minimum 3-connected m-dominating set problem for any positive integer m. While many other efforts are made to design constant factor approximation algorithm for arbitrary k and m positive integer pair, the problem of designing constant factor approximation algorithm for the case with arbitrary $k \geq 4$ and $m \geq 1$ pair is still open [13].

This paper aims to introduce a better approximation algorithm for the minimum 3-connected m-dominating set problem. In detail, we identify the two main drawbacks of the approximation algorithm in [13], [14]: 1) complicated structure of the algorithm, which makes it very difficult to understand and implement; and 2) huge approximation factor. In this paper, we exploit a classical result from graph theory, Tutte decomposition, to design a new approximation algorithm for the problem. The algorithm features with: 1) a drastically simple structure, and 2) a much smaller performance ratio that is nearly half of that of the best existing one. Our simulation results show our new algorithm produces smaller 3-connected m-dominating sets than the competitor on average and coincide with our theoretical performance analysis.

The rest of this paper is organized as follows. Section II introduces notations, definitions, and preliminaries. Related work is discussed in Section III. Our main result, a new constant factor approximation for the minimum 3-connected m-dominating set problem in UDG, and its improved version are proposed in Section IV along with their performance ratio analysis. Our simulation results are presented in Section V. Finally, we conclude this paper in Section VI.

II. Notations, Definitions, and Preliminaries

A. Notations and Definitions

This paper focuses on homogenous wireless networks, i.e., wireless networks of physically equivalent nodes, on two-dimensional euclidean space. In a homogenous wireless network, two nodes u and v have a bidirectional communication link between them only if their distance is close enough. Otherwise, there is no communication link between them. As a result, *unit disk graph* (UDG) is a good graph of choice to abstract the wireless network [10]. Formally, a graph $G = (V, E) = (V(G), E(G))$ on a two-dimensional euclidean space is referred to as a UDG if for each $u, v \in V$, there exists a bidirectional edge between them only if the euclidean distance between them is no greater than 1, i.e., $udist(u, v) \leq 1$. In this paper, for any node subset $V' \subseteq V$, $G[V']$ means a subgraph of G induced by V'. Similarly, for any edge subset $E' \subseteq E$, $G[E']$ will imply a subgraph of G induced by E'. Now, we introduce some important definitions.

Definition 1 [Independent Set (IS) and Maximal IS (MIS)]: Given $G = (V, E)$, a subset $I \subset V$ is an independent set of G if for each pair $u, v \in I$, $(u, v) \notin E$. An independent set I is referred to as a maximal independent set if there exists no node $w \in V \setminus I$ such that $I \bigcup \{w\}$ is still an independent set of G.

Definition 2 [Dominating Set (DS) and Connected DS (CDS)]: Given $G = (V, E)$, a subset $D \subset V$ is a dominating set of G if for each $u \in V$, either $u \in D$, or there exists another node $v \in D$ such that $(u, v) \in E$. A dominating set D, whose induced graph $G[D]$ is connected, is called a connected dominating set.

Definition 3 (k-Vertex-Connectivity): A graph $G = (V, E)$ is k-vertex-connected if for any subset $V' \subset V$ with size at most $k - 1$, $G[V \setminus V']$ is still connected.

For the sake of the simplicity of our discussion, we will use "k-vertex-connected" and "k-connected" interchangeably.

Definition 4 [m-Dominating Set (m-DS)]: Given a graph G and a DS (or CDS) D, D, is a m-dominating set if for any $u \in V \setminus D$, u has at least m neighbors in D.

Definition 5 [k-Connected m-Dominating Set ((k, m)-CDS)]: Given a graph $G = (V, E)$, a subset $C \subseteq V$ is a k-connected m-dominating set of G if: a) $G[C]$ is k-connected, and b) C is a m-dominating set of G.

Definition 6 (Cut-Vertex): Given a graph $G = (V, E)$, a node $u \in V$ is called as a cut-vertex of G if $G[V \setminus \{u\}]$ is disconnected.

Definition 7 (Separator): For a 2-connected graph G, a separator of G is a pair of vertices $\{u, v\} \subseteq V$ such that the subgraph induced by $G \setminus \{u, v\}$ is disconnected.

Definition 8 (H-Path): Given a graph G, an H-path P of a subgraph H is a path between two different nodes in H such that no inner node of P is in H.

The length of an H-path is the number of edges in the path. We use H_k to denote an H-path whose length is no greater than k.

B. Tutte Decomposition of 2-Connected Graphs

Next, we briefly discuss Tutte decomposition [23], which is a classical result in graph theory and plays a vital role in the design and performance analysis of our constant factor approximation algorithm for the minimum 3-connected m-dominating set problem.

Definition 9 (Connected Modulo): Consider a 2-connected graph $G = (V, E)$ and a pair of nodes $\{u, v\} \subset V$. Then, G is a connected modulo with respect to u and v, or in short $[u; v]$, if there are no two nonempty edge subsets E_1 and E_2 such that: a) $E_1 \bigcap E_2 = \emptyset$; b) $E_1 \bigcup E_2 = E(G)$; and c) $G[E_1]$ and $G[E_2]$ intersect only at u and v. Otherwise, G is a nonconnected modulo with respect to u, v, or in short $\neg[u; v]$. Specifically, this is true if:

a) u and v are adjacent; or
b) $V \setminus \{u, v\}$ is disconnected [see Fig. 1(a)].

Definition 10 (Split-Candidate): Consider a separator u, v of a 2-connected graph G. Then, two edge subsets E_1 and E_2 are called a split-candidate of G with respect to u and v, denoted by $[E_1; E_2; u; v]$, if:

a) $G[E_1]$ is a connected module with respect to $[u; v]$; and
b) $G[E_2]$ is 2-connected [see Fig. 1(a)].

In this case, we call $\{u, v\}$ a split-separator.

Based on Tutte's theory [23], it is always possible to decompose a 2-connected graph into a number of 3-connected components with at least 3 vertices, rings, and bonds (two nodes connected by three edges). In this paper, we assume the 2-connected graph has at least 3 edges, otherwise it is trivial. For instance, in the 2-connected graph in Fig. 2(a), there exists a split-candidate with respect to u_1 and u_2, say $S = [E_1; E_2; u_1; u_2]$. Then,

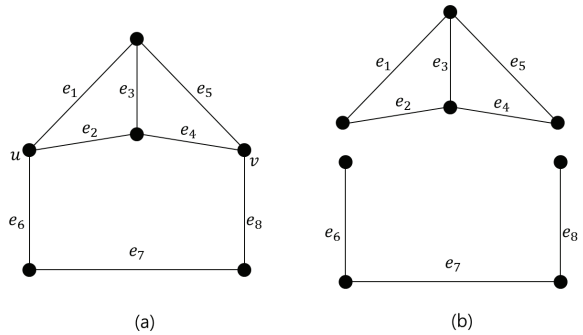


Fig. 1. (a) u and v consist of a separator of the whole graph $G = (V, E)$, which is a nonconnected module with respect to u and v. (b) $[E_1 = \{e_6, e_7, e_8\}; E_2 = \{e_1, e_2, e_3, e_4, e_5\}; u; v]$ is a split-candidate of G since $G[E_1]$ is a connected modulo with respect to u, v and $G[E_2]$ is 2-connected (but not a connected modulo with respect to u, v).

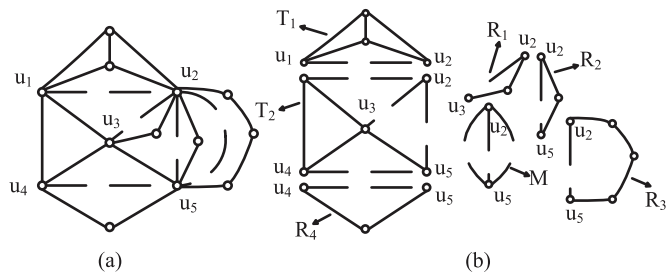


Fig. 2. By Tutte's theory, it is always possible to use the concept of split candidates to completely decompose a 2-connected graph into 3-connected components. In (a), the dot-line is added between the split-separator nodes in the original graph. In (b), the dot-line is a virtual edge, which implies multihop paths between two nodes in the original graph.

from the initial decomposition, we can obtain T_1, which is a connected module with respect to u_1 and u_2, and a 2-connected subgraph, say T'. Once decomposed, we add a virtual edge between u_1 and u_2 in both T_1 and T'. T' is 2-connected, and there is a split-candidate with respect to u_2 and u_5. As a result, we can further decompose the T'. This process is repeated until there is no split-candidate left, e.g., Fig. 2(b) shows the final result of the decomposition process. According to [23], the result of Tutte decomposition is unique, which is independent of the decomposing process. From now on, we will refer to each of the resulting 3-connected components, rings [e.g., the subgraph R_3 in Fig. 2(b)], and bonds [e.g., the subgraph M in Fig. 2(b)] as *T-bricks*, *R-bricks*, and *M-bricks*.

It is known that after Tutte decomposition is completed, there exist at least two virtual edges and at most one edge from the original graph between any pair of nodes. Now, we introduce the concept of the RMT-tree, which can be induced from the result of the decomposition:

1) Each of T-bricks, R-bricks, and M-bricks is contracted into a node in the RMT-tree.
2) There exists an edge between a pair of nodes in the RMT-tree only if their original blocks have virtual edges between the same pair of nodes, e.g., there exists an edge between the node representing T_2 and the node representing R_4 in Fig. 2(b).

Fig. 3 illustrates an example of a resulting RMT-tree from the decomposition result in Fig. 2(b). Let us refer to each node in an RMT-tree by a node as well as a brick (generically, but specifically, T-brick, R-brick, or M-brick). Then, by the definition of the split-candidate and Tutte decomposition's rule, a pair of node $\{u, v\}$ is a separator of G if there exists a split-candidate with respect to u and v during the course of the decomposition process.

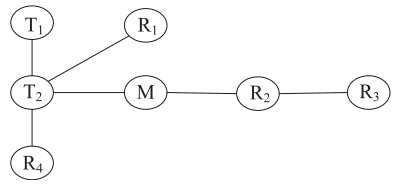


Fig. 3. Resulting RMT-tree of Tutte decomposition of the graph in Fig. 2(a).

C. Some Important Lemmas

Finally, we present some lemmas that are keys to the proof of the performance of the proposed algorithms.

Lemma 1 [13]: Let $C_{2,m}$ be a 2-connected $m(\geq 3)$ dominating set of a 3-connected graph G. Let $\{u, v\}$ be a pair nodes, which is a separator of $C_{2,3}$. Suppose $C_{2,3} \setminus \{u, v\}$ splits into two parts C_1 and C_2. Then, there exists an H-path with length at most three connecting C_1 and C_2 without going through u and v.

By the lemma above, given a $C_{2,3}$ that is not 3-connected, suppose $\{u, v\}$ is a separator of $C_{2,3}$. Then, by adding the internal nodes (the number of which is at most two) of the H-path into $C_{2,3}$, $\{u, v\}$ is not a separator anymore. The key observation is that the newly added nodes do not cause any trouble for us to make $C_{2,3}$ to be 3-connected by introducing any new separator.

Lemma 2: For any 2-connected graph Γ, let x be a new vertex that is adjacent to at least three vertices in Γ, then the graph Γ' obtained from Γ by adding x has no new separator.

Proof: Suppose $\{u, v\}$ is a pair of separators of Γ' but is not a separator of Γ. If one of u or v is x, say $u = x$, then $\Gamma' \setminus \{x, v\} = \Gamma \setminus \{v\}$ is connected since Γ is 2-connected—a contradiction. If neither u nor v is x, note that x has at three neighbors in Γ. If u and v happen to be two of them, the deletion of u and v from Γ' still does not disconnect x from Γ. Thus, $\Gamma' \setminus \{u, v\}$ is still connected—a contradiction. Therefore, $\{u, v\}$ is not a separator of Γ' if $\{u, v\}$ is not a separator of Γ. ■

As a corollary of Lemma 2, we have the following lemma.

Lemma 3: Let $Y = C_{2,3}$ and $\{u, v\}$ be a separator of Y. Let $P = u_0 \ldots v_0$ be the H-path connecting C_1 and C_2. $Y_1 = Y \cup V(P)$, then there are no new separators in Y_1.

Therefore, according to Lemma 3, by successively adding H-paths to remove all the separators, eventually we will make $C_{2,m}$ to be a 3-connected m-dominating set. The key issue concerned is how to guarantee that in this process, the number of the newly added nodes is not too much compared to the size of $C_{2,3}$.

III. Related Work

In [16], Wang *et al.* proposed the first constant factor approximation algorithm for the minimum 2-connected k-dominating set problem in unit disk graph, whose performance ratio

is 64. This algorithm first constructs a connected dominating set using an existing approximation algorithm for the minimum 1-connected 1-dominating set problem and iteratively finds an H-path from a noncut-vertex node in a maximal 2-connected subgraph to another maximal 2-connected subgraph to the connected dominating set until the original dominating set becomes 2-connected by these repeated operations.

In [15], Shang *et al.* proposed a centralized algorithm to construct 2-connected m-dominating set. Given a unit disk graph $G = (V, E)$, the algorithm first computes an 1-connected m-dominating set C and merges this with a series of independent sets $I_2, I_2, \ldots, I_m$ by repeatedly computing an independent set I_i from the residual graph $G[V \setminus (C \bigcup I_1 \bigcup \cdots \bigcup I_{i-1})]$ for each $i = 2, \ldots, m$ in order, where $I_1 = \emptyset$. Once the 1-connected m-dominating set is computed, an idea similar to [16], but only using H-paths with length at most 3, is applied to make the 2-connected m-dominating set. In [22], Shi *et al.* further gave an improved approximation algorithm for computing a $(2, m)$-CDS based on a greedy strategy.

Over the years, several efforts were made to design a constant factor approximation algorithm for the k-connected m-dominating set problem in unit disk graph for arbitrary integer k and m pairs [17]–[21]. However, all of them do not work correctly in a sense that the approximation bound is incorrect (and cannot be bounded) or the algorithm does not produce a feasible solution for some graph instances despite of their existence [13]. Very recently, in [14], Wang *et al.* have proposed a new constant factor approximation algorithm for the minimum 3-connected m-dominating set problem in unit disk graph for any positive integer m. However, the design and implementation of the algorithm is nontrivial, and the constant approximation ratio is huge. In this paper, we aim to introduce a simpler approximation algorithm, which is easier to implement and is with a smaller performance ratio, for the minimum 3-connected m-dominating set problem by using Tutte's decomposition technique.

IV. Better Constant Factor Approximation for Minimum 3-Connected m-Dominating Set Problem in UDG

A. Basic Algorithm

In this section, we introduce our new constant factor approximation algorithm for the minimum 3-connected m-dominating set problem in unit disk graph based on the theory of Tutte decomposition. In Section IV-B, we introduce an enhanced version of this algorithm, which is with a simpler algorithmic structure and is with a smaller performance ratio.

The main idea of the algorithm is as follows. First, we compute a $(2, m)$-CDS using the algorithm in [22] (line 1 of Algorithm 1), and decompose the $(2, m)$-CDS into T-bricks, R-bricks, and M-bricks (line 1 of Algorithm 1). By the procedure of Tutte decomposition, it is known that M-bricks that are formed by several edges between two nodes has no effect on the result of our algorithm. Therefore, we only take T-bricks and R-bricks into our consideration. Next, for every R-brick with more than three nodes, we try to eliminate all of its "local" separators (i.e., the separators $\{u, v\}$ with $u, v \in V(R)$, which are not shared by any pair of bricks) by adding some H-paths successively (lines 3–8 of Algorithm 1). After this step is completed, the remaining separators (if there are any) are only some of the split-separators. At last, removing all split-separators sequentially based on the RMT-tree structure (lines 9–12 of Algorithm 1) makes the final $C_{2,m}$ to be a $C_{3,m}$, for $m \geq 3$.

Algorithm 1 Tutte-3-m-CDS ($G = (V, E)$)

1: Compute a 2-connected m-dominating set Y by using an existing r-approximation algorithm for some positive constant r.
2: Tutte decomposes Y and obtains the set Y_1 of R-bricks such that the cardinality of each brick is at least 4.
3: **for** each R-bricks Y' in Y_1 **do**
4: **while** there exists a separator $\{u, v\}$ of Y on a ring $R \in Y_1$ such that u and v are not a split-separator (that is a separator shared by two bricks of any types). **do**
5: Find an H_3 path from a node $w \in R \setminus \{u, v\}$ to $V \setminus R$ such that $\{u, v\}$ is not a separate anymore on the induced graph by $G[Y \bigcup V(H_3)]$.
6: Set $Y \leftarrow Y \bigcup V(H_3)$.
7: **end while**
8: **end for**
9: **while** there exists a separator $\{u, v\}$ of Y such that u and v are shared by two bricks of any types. **do**
10: Find an H_3 path from a node $w \in R \setminus \{u, v\}$ to $V \setminus R$ such that $\{u, v\}$ is not a separate anymore on the induced graph by $G[Y \bigcup V(H_3)]$.
11: Set $Y \leftarrow Y \bigcup V(H_3)$.
12: **end while**
13: Output Y.

Theorem 1: The output of Algorithm 1 is a 3-connected m-dominating ($m \geq 3$) set.

Proof: Let $\{u, v\}$ be any separator in $C_{2,m}$. Then, according to the theory of Tutte decomposition, $\{u, v\}$ either belongs to an R-brick or is shared by two bricks. In the former case, by lines 3–8 of Algorithm 1, all separators will be removed; in the later case, by lines 9–12 of Algorithm 1, $\{u, v\}$ will be removed eventually. Thus, $\{u, v\}$ will be eliminated after the execution of the Algorithm 1. Moreover, by Lemma 3, the process of removing separators does not introduce any new separators. Thus, at the end of Algorithm 1, all separators are removed, and we get a 3-connected graph. ■

Theorem 2 [22]: The algorithm for the minimum $(2, m)$-CDS has a performance ratio $\alpha + 2(1 + \ln \alpha)$ for $m > 3$, where α is the approximation ratio for the minimum $(1, m)$-CDS and 12.46 for $m = 3$.

The following lemma lies at the heart of our analysis of Algorithm 1, which is proved by induction. An alternative and more straightforward proof will be given in Section IV-B.

Lemma 4: Let $C_{2,m} (m \geq 3)$ be a cycle on n nodes, and $T(n)$ be the number of H_3-paths needed to remove all the separators of $C_{2,m}$. Then, we have $T(n) \leq 2n - 6$.

Proof: We prove the lemma by induction on the number of nodes n. When $n = 4$, there are at most two separators in total,

therefore at most $2 \times 4 - 6 = 2$ H_3-paths are needed to remove all the separators. Assuming the lemma is correct for any cycle with nodes less than or equal to $n-1$, we show the lemma is true for n.

Note that after an H_3-path $P = u_0 \dots v_0$ has been added, the cycle is split into two cycles (the effect of adding an H_3-path is equivalent to adding a virtual edge u_0v_0), namely, R_1 and R_2, where $R_1 \cap R_2 = \{u_0, v_0\}$ and $|R_1| = n_1 < n$, $|R_1| = n_2 < n$, $n_1 + n_2 = n + 2$. The key observation is that after an H_3-path P has been added to $C_{2,m}$ (not considering the inner nodes of H_3-path), the pair of nodes $\{u, v\}$ that is used to be a separator of $C_{2,m}$ is not a separator of $C_{2,m}$ anymore, $\forall u \in R_1 \setminus \{u_0, v_0\}$, $\forall v \in R_2 \setminus \{u_0, v_0\}$. Thus, after the first H_3-path P is added, all separators of $C_{2,3} \cup V(P)$ are now contained in R_1 and R_2, respectively.

The process of adding H_3-paths can proceed whenever there are separators left. Note that when we are trying to remove a separator in R_1 (resp. R_2) by adding an H_3-path P', it may happen that some separators in R_2 (resp. R_1) have been removed at the same time because of the adding of P'. Of course, this may reduce the total number of the H_3-paths needed to remove all the separators of $C_{2,m}$. Thus, in the most unfavorable conditions, adding H_3-paths to remove separators in R_1 (resp. R_2) does not help removing the separators in R_2 (resp. R_1), and hence the total number of the H_3-paths needed to eliminate all separators in $C_{2,m}$ is at most the number of H_3-paths needed to remove all separators in R_1 and R_2 separately, plus two. That is, $T(n) \le T(n_1)+T(n_2)+1+1 \le (2n_1-6)+(2n_2-6)+2 = 2(n_1+n_2)-10 = 2n-6$, where we have used the induction hypothesis that $T(n_1) \le 2n_1 - 6$ and $T(n_2) \le 2n_2 - 6$, and two additional H_3-paths are added—the first one is P and the second one is the H_3-path that might be needed to remove the separator $\{u_0, v_0\}$. ■

When we add H_3-path to connect the components separated by $\{u, v\}$ that is on R-brick R_1 and not a split-separator, the endpoints of the H_3-path may be in other bricks. However, once the H_3-path has been added, we always can find equivalent (will be discussed in more detail in Section IV) H_3-paths whose endpoints are on R_1, so the above lemma still stands.

Theorem 3: Algorithm 1 is an 87.23-approximation for computing (3,3)-CDS, where $r = 12.46$ is the approximation ratio for computing a $(2, m)$-CDS in unit disk graphs.

Proof: We have an r-approximation algorithm for computing a (2,3)-CDS with $r = 12.46$ (see [22]). Thus, we can get a $C_{2,3}$ such that $|C_{2,3}| \le r|C_{2,3}^{\text{opt}}|$, where $C_{2,3}^{\text{opt}}$ is the optimal solution of the 2-connected 3-dominating set problem. Let S_1 denote the set of the H_3-paths needed to eliminate the separators of all the R-bricks, and S_2 denote the set of the split-separator. Suppose that $R = \{R_1, R_2, \dots, R_s\}$ is the set of R-bricks, and $|R_i| = n_i$. According to Lemma 4, $|S_1| \le \sum_{i=1}^{s}(2n_i - 6) \le 2|C_{2,3}|$. Clearly, every two adjacent bricks share two nodes that consist of a split-separator, so there are $|C_{2,3}|$ bricks, and we have $|S_2| \le |C_{2,3}|$. The output $C_{3,3} = C_{2,3} \cup H$, where H denotes the set of all nodes added. To estimate the performance ratio of Algorithm 1, according to Lemmas 1 and 4, we have $|C_{3,3}| = |C_{2,3} \cup H| = |C_{2,3}| + |H| \le |C_{2,3}| + 2|S_1| + 2|S_2| \le |C_{2,3}| + 4|C_{2,3}| + 2|C_{2,3}| \le 7r|C_{2,3}^{\text{opt}}| \le 7r|C_{3,3}^{\text{opt}}| = 87.23|C_{3,3}^{\text{opt}}|$. This completes the proof. ■

Algorithm 2 Simple-3-m-CDS ($G = (V, E)$)

1: Compute a 2-connected m-dominating set D by using an existing r-approximation algorithm for some positive constant r.
2: **while** There exists a separator $\{u, v\}$ in D **do**
3: By Definition, $G[D \setminus \{u, v\}]$ is split into two subgraphs $G[D_L]$ and $G[D_R]$. Find an H_3-path from a node in D_L to another node in D_R, and add the nodes on the H_3 path to D.
4: **end while**
5: Return D.

Similarly, we have the following theorem.

Theorem 4: Algorithm 1 is a $7r$-approximation for computing $(3, m)$-CDS, for $m > 3$, where r is the approximation ratio of the minimum $(2, m)$-CDS in [22].

Theorem 5: Let n be the number of nodes in the original graph. Then, the time complexity of Algorithm 1 is $O(n^4)$.

Proof: Computing a $(2, m)$-CDS takes time $O(n^3)$. The unique decomposition of a 2-connected graph can be found in linear time; see [28]. The time complexity of second step and third step is dominated by the Shortest-Path function, which runs in $O(n^2)$. The if-loop is executed at most $O(n^2)$, so the process of adding H_3-path takes time $O(n^4)$. The second and third steps are executed at most n times since the number of the bricks is at most n. Therefore, the time complexity is at most $O(n^4)$. ■

B. Simplified Algorithm

In this section, we give a much simpler approximation algorithm with smaller performance ratio. We show that the naive idea of "adding H_3-paths to remove separators successively until no separator left" is actually a constant approximation (see Algorithm 2 for details). The prominent difference of the second algorithm from the previous one is that we only use Tutte decomposition in the performance analysis of our algorithm.

First, we focus on the special case that $C_{2,m}$ happens to be a cycle C_n on n vertices, and then we show the general case can actually be reduced to this special one, by using Tutte decomposition.

For better visualization, next we can use a convex polygon Ω with n vertices to represent the cycle C_n, and we also use a line segment to denote an H_3-path with the same endpoints. Label the vertices of polygon clockwise, and denote it by its vertex set $\{v_1, v_2, \dots, v_n\}$. Clearly, $\{v_i, v_j\}$ is a separator for all nonadjacent vertices v_i and v_j. Thus, there are $\binom{n}{2} - n = O(n^2)$ pairs of separators in total. However, we are able to show that at most $O(n)$ H_3-paths (rather than $O(n^2)$ H_3-paths) are needed to eliminate all separators, and hence to make C_n to be 3-connected. As aforementioned, this is due to the important fact that when we add one H_3-path to remove a specific pair of separators, we actually remove many separators at the same time.

More precisely, at the beginning, we have a $C_{2,m}(m \ge 3)$ that is represented by a convex n-polygon Ω on n vertices.

Suppose $\{u, v\}$ is a separator of C_n, and $C_n \setminus \{u, v\}$ is decomposed into of two parts C_L and C_R. Now add an H_3-path with two ending points $s_1 \in C_L$ and $t_1 \in C_R$ connecting C_L and C_R. After adding the H_3-path $P_{s_1t_1}$, $\{u, v\}$ is no longer a separator. More importantly, any vertex in $C_L \setminus \{s_1, t_1\}$ and any vertex in $C_R \setminus \{s_1, t_1\}$ cannot constitute a pair of separators anymore. Put another way, after adding $P_{s_1t_1}$ (we may think of it as a line segment), the convex polygon Ω is partitioned into two smaller subpolygons: Ω_1 and Ω_2, and any pair of separators $\{u, v\}$ in Ω must satisfy $u, v \in \Omega_1$ or $u, v \in \Omega_2$ after adding $P_{s_1t_1}$.

To make our presentation more concise, we introduce the following.

Definition 11: Let $\{u, v\}$ be a pair of separators. Let P be an H_3-path newly added. If after adding of P, $\{u, v\}$ is not a separator anymore, we call the H_3-path P crosses or removes the separator $\{u, v\}$.

Definition 12: We call an H_3-path crosses a convex polygon Ω if after adding the H_3-path, some pair of separators $\{u, v\}$, $u, v \in \Omega$ are removed.

In order to give an upper bound for the number of H_3-paths needed to remove all separators in Ω, we take a copy of Ω, denoted by Ω'. Later, we will show the number of H_3-paths needed to remove all separators in Ω is actually bounded by that for Ω'. Whenever an H_3-path is added in Ω, a set of corresponding H_3-paths (called virtual paths) is added in Ω' according to some specific rules described below. Notice that the paths added in Ω can intersect each other (we may think of the H_3-path as a line segment with the same endpoints as the H_3-path), but the virtual paths never intersect each other (except that they can share an endpoint in common). Thus, the virtual paths in Ω' induce a series of partitions of Ω' into smaller subpolygons.

First, when $P_1 = s_1t_1$ is added in Ω, the corresponding virtual paths added in Ω' are $P'_1 = P_1 = s_1t_1$. The adding of P'_1 partitions Ω' into two subpolygons Ω'_1 and Ω'_2.

Next, we add $P_2 = s_2t_2$ into Ω. There are two possible cases: 1) s_2 and t_2 are contained in the same subpolygons, say Ω'_1. Then, $P'_2 = P_2 = s_2t_2$; 2) s_2 and t_2 are contained in different subpolygons, say $s_2 \in \Omega'_1$ and $t_2 \in \Omega'_2$. In this case, $P_2 = s_2t_2$ crosses with Ω'_1 and Ω'_2. Then, we add virtual paths s_2t_1, s_2s_1, t_2t_1, and t_2s_1, provided they are diagonals of Ω' (by a diagonal of polygon, we mean the line segment connecting two nonadjacent vertices).

Generally, suppose we have added a list of H_3-paths $P_1, P_2, \ldots, P_{i-1}$ sequentially into Ω, and the corresponding virtual paths $P'_1, P'_2, \ldots, P'_{m_i}$ partitioned Ω' into subpolygons $\Omega_i^{(1)}, \Omega_i^{(2)}, \ldots, \Omega_i^{(l_i)}$. Then, when adding the ith H_3-path $P_i = s_it_i$ into Ω, the corresponding virtual paths are added in Ω' according to the following rules.

- *Rule 1*: Let $\Omega_i^{(j)}$ be any subpolygon, for $j = 1, 2, \ldots, l_i$. Suppose that $\Omega_i^{(j)}$ is not a triangle. If the H_3-path $P = s_it_i$ does not cross with $\Omega_i^{(j)}$, then no virtual paths are added in $\Omega_i^{(j)}$. Otherwise, suppose the H_3-path $P = s_it_i$ crosses with $\Omega_i^{(j)}$, then the line segment $\overline{s_it_i}$ intersects with the convex subpolygon $\Omega_i^{(j)}$ at exactly two edges, say $\overline{u_0v_0}$ and $\overline{u_1v_1}$, of $\Omega_i^{(j)}$, where u_0 and v_0 (resp. u_1 and v_1) are separated by the line s_it_i (i.e., they lie at different parts of the plane partitioned by s_it_i), and u_0 and u_1 (resp. v_0 and v_1) are not separated by s_it_i.

  Let $\overline{u_0v_0} \cap \overline{s_it_i} = \{x\}$ and $\overline{u_1v_1} \cap \overline{s_it_i} = \{y\}$. We distinguish three cases.

  Case 1: First, suppose that $x \notin \{u_0, v_0\}$ and $y \notin \{u_1, v_1\}$. If $\overline{u_0u_1}$ (resp. $\overline{v_0v_1}$) is a diagonal of $\Omega_i^{(j)}$, then add u_0u_1 (resp. v_0 and v_1) as virtual edges in Ω' that partition $\Omega_i^{(j)}$ into three smaller subpolygons, namely, T_1, T_2, and T_3, where T_1 and T_2 share two nodes u_0, u_1 in common; T_2 and T_3 share two nodes v_0, v_1 in common; and $T_2 = \{u_0, v_0, v_1, u_1\}$ is a quadrangle with four nodes. Now adding either a virtual edge u_0v_1 or u_1v_0 (but not both), T_2 is split into two triangles $T_{21} = \{u_0, v_0, v_1\}$ and $T_{22} = \{u_0, v_1, u_1\}$ (or $\{u_0, v_0, u_1\}$ and $\{u_1, v_1, v_0\}$; see Fig. 4 for example.

  Case 2: $x \in \{u_0, v_0\}$ (say $x = u_0$) but $\{y\} \notin \{u_1, v_1\}$. If u_0u_1 (resp. u_0v_1) is a diagonal of $\Omega_i^{(j)}$, then add virtual edge u_0u_1 (resp. u_0v_1). Then, $\Omega_i^{(j)}$ was partitioned into three subpolygons; see Fig. 5 for example.

  Case 3: $x \in \{u_0, v_0\}$ (say $x = u_0$) and $y \in \{u_1, v_1\}$ (say $y = u_1$). In this case, u_0u_1 is always a diagonal of $\Omega_i^{(j)}$, and the virtual edge added in $\Omega_i^{(j)}$ coincides with $P = s_it_i$ that partitions $\Omega_i^{(j)}$ into two smaller subpolygons; see Fig. 6 for example.
- *Rule 2*: If $\Omega_i^{(j)}$ is a triangle (no matter H_3-path $P_{s_it_i}$ crosses with $\Omega_i^{(j)}$ or not), then $\Omega_i^{(j)}$ remains unchanged, and no virtual path corresponding this subpolygon will be added in Ω'.

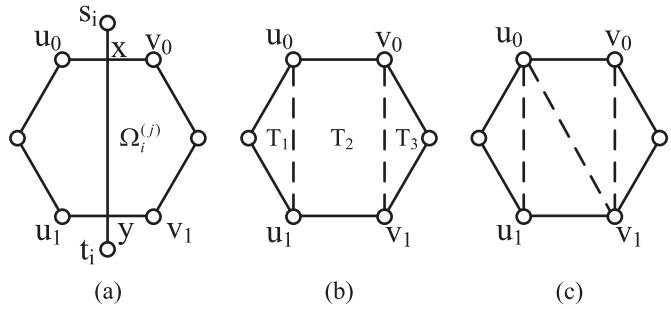


Fig. 4. (a) $P_{s_it_i}$ is the H_3-path added into Ω, and s_it_i crossed the two edges u_0v_0 and u_1v_1. (b) Virtual edges u_0u_1 and v_0v_1 partition $\Omega_i^{(j)}$ into three smaller subpolygons, namely, T_1, T_2, and T_3. (c) Subpolygon $\{u_0, v_0, v_1, u_1\}$ continues to split into $\{u_0, v_0, v_1\}$ and $\{v_1, u_1, u_0\}$ by u_0v_1.

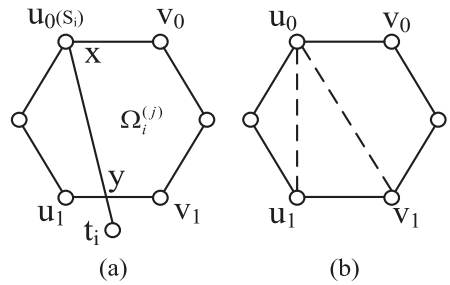


Fig. 5. (a) H_3-path $P_{s_it_i}$ added into Ω, s_it_i crossed the edge u_1v_1 and $s_i = u_0$. (b) By adding virtual paths u_0u_1 and u_0v_1 sequentially, $\Omega_i^{(j)}$ has been subdivided into three subpolygons.

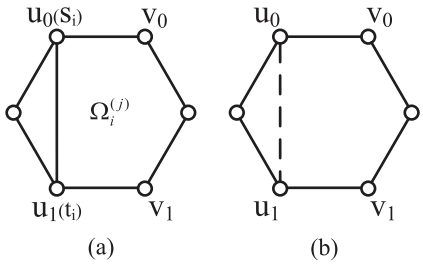


Fig. 6. (a) Endpoints of the H_3-path $P_{s_it_i}$ added are in $\Omega_i^{(j)}$. (b) Virtual edge added in $\Omega_i^{(j)}$ is s_it_i itself, which can partition the polygon into two subpolygons.

The above process can go on, until Ω' has been triangulated (i.e., all subpolygons are triangles). Then, while we may still need to add H_3-paths in Ω to remove some separators (if there is any), no corresponding virtual edges will be added in Ω' (since it has been triangulated). An example is given in Fig. 7 to illustrate of the above process.

Now we give an important property about the above constructions.

Lemma 5: Let $\widetilde{S}_i^1$ denote the set of separators removed by adding the H_3-path $\mathcal{P}_i = \{P_1, P_2, \ldots, P_i\}$ sequentially into Ω, and $\widetilde{S}_i^2$ denotes the set of the separators in Ω' removed by the virtual paths corresponding to $\mathcal{P}_i$ using the above rules. Then, we have $\widetilde{S}_i^1 \supset \widetilde{S}_i^2$.

Proof: Let $P_1, P_2, \ldots, P_k$ be the H_3-paths sequentially added such that all separators of $\Omega = C_{2,m}$ have been removed. During the process of adding these H_3-paths, the corresponding virtual H_3-paths are also added in the convex polygon Ω' according the rules described above.

Suppose that when adding $P_1, P_2, \ldots, P_s$, the polygon Ω' has been partitioned into triangles by the corresponding virtual H_3-paths $P_1', P_2', \ldots, P_t'$. P_{s+1} is the first H_3-path added, corresponding to which there are no virtual H_3-paths added anymore (since Ω' has been triangulated).

For $i > s$, there are no corresponding virtual edges, so we just need to show the lemma to be true for $i \leq s$.

Next, we prove the lemma is true for $1 \leq i \leq s$ by induction on i. The base case $i = 1$ is obviously true. Supposing the lemma is true for $\mathcal{P}_{i-1} = \{P_1, P_2, \ldots, P_{i-1}\}$, we show the lemma is true for $\mathcal{P}_i$.

Suppose that after adding the virtual paths corresponding to $P_1, P_2, \ldots, P_{i-1}$ in Ω', Ω' has been partitioned into subpolygons $\Omega_i^{(j)}$, for $j = 1, 2, \ldots, l_i$. Notice that all the pairs of separators of Ω_i' are $\cup_{j=1}^{l_i} S(\Omega_i^{(j)})$ (where $S(\Omega_i^{(j)})$ denotes the set of separators $\{x, y\}$ with $x, y \in \Omega_i^{(j)}$). By induction hypothesis, $\widetilde{S}_{i-1}^1 \supset \widetilde{S}_{i-1}^2$. It follows that all the remaining pairs of separators of Ω are included in $\cup_{j=1}^{l_i} S(\Omega_i^{(j)})$, after adding the H_3-paths $P_1, P_2, \ldots, P_{i-1}$ in Ω.

Now consider the rules of adding virtual paths in Ω' corresponding P_i. Suppose Rule 1 Case 1 is true (other cases can be proved in a similar way), then $\Omega_i^{(j)}$ is partitioned into four subpolygons T_1, T_3 and T_{21}, T_{22} by adding three virtual paths, say u_0u_1, u_0v_1 and v_0v_1 (see Fig. 6). Now consider the node set corresponding the node set of $\Omega_i^{(j)}$ in Ω (we use the same notation), then in Ω, all possible pairs of separators are in $S(\Omega_i^{(j)})$ (some of them may not be separators in Ω) before adding P_i. Next, we add P_i into Ω. Let $C_1 = \{u_0, \ldots, u_1\}$ and $C_2 =$

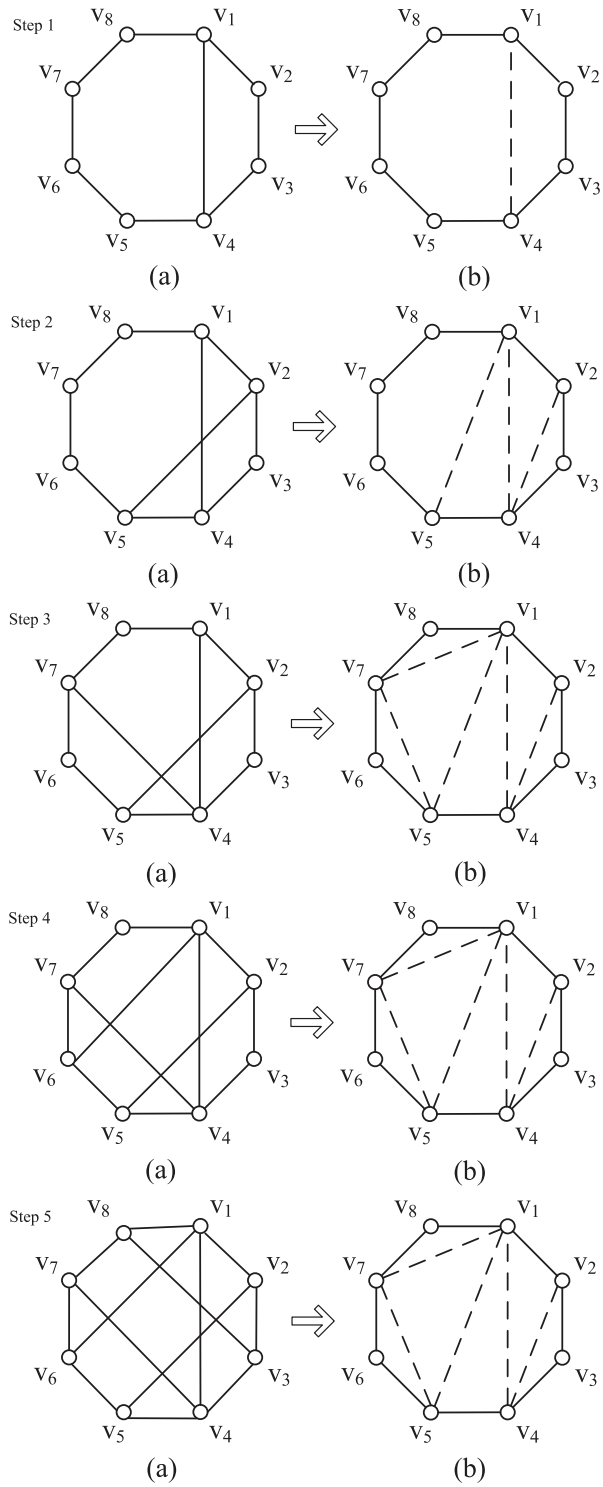


Fig. 7. (a) Process of adding an H_3-path. (b) Process of adding virtual paths in Ω'. Step 1: P_{14} is added in Ω, and since the endpoints of P_{14} are in same polygon $\{v_1, \ldots, v_8\}$, (a) and (b) are same. Step 2: (a) H_3-path P_{25} is added into Ω; (b): its corresponding virtual paths are P_{15} and P_{24} added into Ω_1', Ω_2'. Step 3: (a): H_3-path P_{47} has crossed two polygons, one of which has 3 nodes, so it cannot split into smaller polygons. The other polygon $\{v_1, v_5, \ldots, v_8\}$ can split into three polygons by P_{17} and P_{57}. Since after Step 3 $\{v_1, \ldots, v_8\}$ have been split into triangles, in Steps 4 and 5, the H_3-paths P_{25} and P_{38} are added to remove the separators shared by the triangles, however Ω' cannot split into smaller subpolygons.

$\{v_0, \ldots, v_1\}$ be the partition of the node set of $\Omega_i^{(j)}$. Then, all pairs of separators of Ω in $S(\Omega_i^{(j)})$ are now restricted to $S(C_1) \cup S(C_2)$. On the other hand, all the separators of $\Omega_i^{(j)}$ are $S(T_1) \cup S(T_2) \cup (S(T_{21}) \cup S(T_{22}))$, after adding the virtual paths corresponding to P_i. Clearly, $S(C_1) \cup S(C_2) \subset S(T_1) \cup S(T_2) \cup (S(T_{21}) \cup S(T_{22}))$ (they actually differ by the pairs of separators consisting of the two endpoints of all virtual paths). Note that $\Omega_i^{(j)}$ is any subpolygon of Ω'. If we consider all subpolygons, we know that after adding P_i and the corresponding virtual paths, the remaining set of separators of Ω is contained in that of Ω', i.e., $\widetilde{S}_i^1 \supset \widetilde{S}_i^2$ holds for i. This completes the proof. ■

Lemma 6: Let Ω be a convex polygon on $n(\geq 4)$ vertices. Then, Ω can be partitioned into $n-2$ triangles by adding $n-3$ diagonals, no matter how we partition the polygon.

Proof: We prove the lemma by induction on n. When $n = 4$, the lemma is obviously true. Assume it is true for a convex polygon with less than or equal to $n-1$ vertices. Then, adding one diagonal arbitrarily partitions Ω into two subpolygons Ω_1 and Ω_2 with n_1 and n_2 vertices, respectively. By induction hypothesis, the lemma is true for Ω_1 and Ω_2, i.e., Ω_i can be partitioned into $n_i - 2$ triangles by adding $n_i - 3$ diagonals, for $i = 1, 2$. Thus, Ω can be partitioned into $(n_1 - 2) + (n_2 - 2) = n - 2$ triangles by adding $(n_1 - 3) + (n_2 - 3) + 1 = n - 3$ diagonals. ■

Lemma 7: Let $C_{2,m}$ be a cycle on n nodes. Then, at most $2n - 6$ $(n \geq 3)$ H_3-paths are needed to remove all separators of $C_{2,m}$ and make it to be 3-connected.

Proof: Let $P_1, P_2, \ldots, P_k$ be the H_3-paths sequentially added such that all separators of $\Omega = C_{2,m}$ have been removed. During the process of adding these H_3-paths, the corresponding virtual H_3-paths are also added in the convex polygon Ω' according the rules described above.

Suppose that when adding $P_1, P_2, \ldots, P_s$, the polygon Ω' has been partitioned into triangles by the corresponding virtual H_3-paths $P_1', P_2', \ldots, P_t'$. P_{s+1} is the first H_3-path added, corresponding to which there is no virtual H_3-paths added anymore (since Ω' has been triangulated). According to Lemma 6, an n-polygon can be partitioned into $n-2$ triangles by adding $n-3$ diagonals, which is independent of the ways of the triangulation. Moreover, for each of $P_1, P_2, \ldots, P_s$, the number of corresponding virtual H_3-paths added is at least one. It follows that $s \leq t = n - 3$.

After we have added the H_3-paths $P_1, P_2, \ldots, P_s$ in Ω, the adding of the corresponding virtual H_3-paths partitioned Ω' into triangles. Notice now in Ω', all possible separators are the pairs of vertices that are the two endpoints of the virtual H_3-paths $P_1', P_2', \ldots, P_t'$. According to Lemma 5, now all the possible separators in Ω are also the pair of two endpoints of $P_1', P_2', \ldots, P_t'$. Additional H_3-paths $P_{s+1}, \ldots, P_k$ are added to remove these separators. Thus, in the worst cases, we need another $n-3$ H_3-paths. Therefore in total, the number of H_3-paths needed is at most $2(n-3) = 2n - 6$. ■

Theorem 6: The output of Algorithm 2 is a 3-connected m-dominating $(m \geq 3)$.

Proof: Because of the termination condition of the algorithm, all separators have been removed, and moreover, by Lemma 1, in the process, no new separators will be introduced. Thus, the output of Algorithm 2 is 3-connected. ■

In the previous discussions, we regard the H_3-path as a line segment and ignore its internal nodes. This is not a serious restriction since, according to Lemma 2, the internal nodes can only help us to make $C_{2,m}$ to be 3-connected and do not make any troubles. If we have considered the effect of internal nodes, the total number of H_3-paths needed to removed all separators of $C_{2,m}$ would be further reduced.

Also, we implicitly assume that the two endpoints of the H_3-path added are in $C_{2,3}$. This is also not a serious restriction, as the following lemma shows that we can always choose some nodes in $C_{2,3}$ to act the same role. The following lemma makes sure wherever the endpoints of the last H_3-path are, we always can find the nodes on $C_{2,3}$ to act as the endpoints.

Lemma 8: If the endpoints of the newly added H_3-path are the intermediate nodes of the H_3-path added before, we can find existing nodes in $C_{2,3}$ that can be acted as the endpoints of the H_3-path.

Proof: Since the new added nodes are dominated by at least three nodes in $C_{2,m}$ at least, the lemma is true. ■

Next, we consider the case that $C_{2,3}$ is a general 2-connected graph. The main idea is similar to that of the special case before. The main difference is that we have to use Tutte decomposition for our analysis.

Definition 13: If the separators in the brick are no longer separators after adding the H_3-path, we call the H_3-path crosses the brick.

From Tutte decomposition theory, every general 2-connected graph can be decomposed into bricks and the bricks form a unique tree. Therefore, we have $C_{2,3} = B_1 \cup B_2 \cup \cdots \cup B_r$ with B_i being T-Bricks or R-Bricks. We know adding an H_3-path $P_{u_0v_0}$ can at least reduce one separator. Now suppose the endpoints of the added H_3-path are in two distinct bricks, say in B_1 and B_2 and $u_0 \in B_1$ and $v_0 \in B_2$. If the H_3-path $P_{u_0v_0}$ crosses with the separators in any other blocks B_j, then adding H_3-path also reduces the number of separators in B_j. Thus, for each B_j, the effect of adding the H_3-path $P_{u_0v_0}$ can be replaced by adding other virtual H_3-paths $P_{u_jv_j}$ whose endpoints are in same brick; see Figs. 8 and 9 for example. There are three rules for how to choose the endpoints of the virtual H_3-path.

- *Rule 1*: H_3-path $P_{u_0v_0}$ crosses two bricks B_1 and B_2 and $u_0 \in B_1$ and $v_0 \in B_2$, $B_1 \cap B_2 = \{u_1, v_1\}$. Obviously, $\{u_1, v_1\}$ is a split-separator. For B_1, we add the virtual paths u_0u_1 and u_0v_1 sequentially, if there are no edges between u_0 and $x \in \{u_1, v_1\}$ in R_1 (the virtual edge that is introduced by the Tutte decomposition is treated as an existing edge in the brick). For B_2, it is similar. Hence, bricks B_1 and B_2 can be split after adding $P_{u_0v_0}$; see Fig. 8.
- *Rule 2*: H_3-path $P_{u_0v_0}$ crosses more than two bricks (we take the example of three bricks into consideration) and $u_0 \in B_1$ and $v_0 \in B_2$, $B_1 \cap B_3 = \{u_1, v_1\}$, $B_2 \cap B_3 = \{u_2, v_2\}$. $\{u_1, v_1\}$ and $\{u_2, v_2\}$ are split-separators. For B_1 and B_2, it is the same as Rule 1. Furthermore, for B_3, we choose $x \in \{u_1, v_1\}$ to be one endpoint of the first virtual path, and $y \in \{u_2, v_2\}$ to be the other one endpoint if there is no edge between x and y. Then, B_3 can be split into two smaller polygons T_{31} and T_{32}. If $x' \in \{u_1, v_1\}$ and $y' \in \{u_1, v_1\}$ are in same polygon T_{31} or T_{32} and there exists no edge between them, then T_{31} and T_{32} continue to

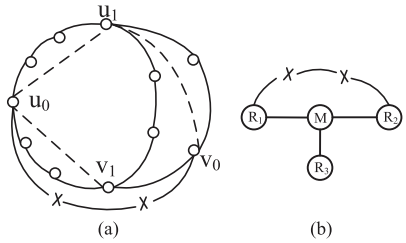


Fig. 8. (a) H_3-path $P_{u_0v_0}$ crosses bricks B_1 and B_2. $\{u_1, v_1\}$ is the split-separator shared by B_1 and B_2. (b) For the R-brick R_1, the virtual paths $P_{u_0u_1}$ and $P_{u_0v_1}$ can make $R_1 = \{u_1, \ldots, v_1\}$ split into subpolygons $\{v_1, \ldots, u_0\}$, $\{u_0, \ldots, u_1\}$, and $\{u_1, v_1, u_0\}$. For the R-brick R_2, H_3-path $P_{u_0v_0}$ is corresponding to the virtual path $P_{u_1v_0}$, and we cannot choose v_1 to be the endpoint because it is adjacent to v_0.

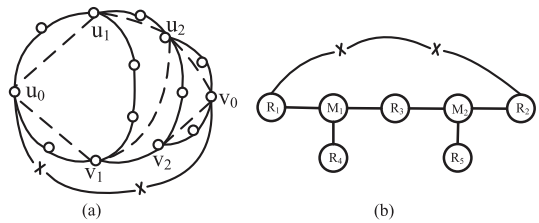


Fig. 9. (a) H_3-path $P_{u_0v_0}$ crosses bricks B_1 B_2 and B_3. $\{u_1, v_1\}$ and $\{u_2, v_2\}$ are the split-separators shared by them. (b) For R_1, it will be split by virtual paths $P_{u_0u_1}$ and $P_{u_0v_1}$; for R_2, it can be split by $P_{u_2v_0}$ and $P_{v_2v_0}$; for B_3, $P_{v_2v_1}$ and $P_{u_1v_2}$ will make it split into smaller polygons.

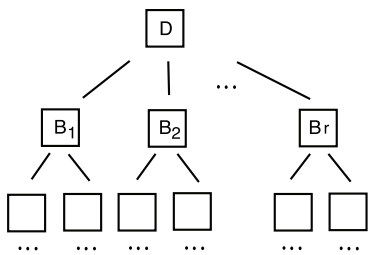


Fig. 10. General 2-connected graph based on Tutte decomposition, and all the R-bricks can be split into smaller polygons.

split by the virtual path $P_{x'y'}$ until any $x \in \{u_1, v_1\}$ and $y \in \{u_1, v_1\}$ in same polygon there is an edge; see Fig. 9.

- *Rule 3*: When s_i or t_i is in the bricks that has three nodes or is in a T-brick, the bricks remain unchanged.

From above discussions, whenever we add an H_3-path $P_{u_0v_0}$ to remove a separator $\{u, v\}$ in a R-brick, and whenever there are some separators in B_i crosses with $P_{u_0v_0}$, we can find a virtual path $P_{u_iv_i}$ in B_i to make it split. Thus, whenever there is an H_3-path added, the R-bricks can be split according to similar rules as previously; see Fig. 10 for an illustration. Moreover, note there are no separators in T-bricks (except the splitting separator); we can consider T-bricks as R-bricks with three nodes without effecting our results.

Lemma 9: $\widetilde{S}_1$ denotes the set of the removed separators by the H_3-path $P_{u_0v_0}$ added in reality, and $\widetilde{S}_2$ denotes the set of the removed separators by the virtual paths corresponding to $P_{u_0v_0}$ as the above rule, $\widetilde{S}_1 \supset \widetilde{S}_2$.

Proof: The proof is similar to Lemma 5. ■

Lemma 10: In Algorithm 2, at most $2n - 6$ H_3-paths are needed to make $C_{2,3}$ be 3-connected, where $n := |C_{2,3}|$.

Proof: Suppose that $D = C_{2,3}$ is decomposed into bricks $B_1, B_2, \ldots, B_r$ $(1 \le r)$ with $|D| = n$ and $|B_i| = n_i$. According to Tutte decomposition, we have $\sum_{i=1}^{r} n_i = n + (r-1) \times 2$ (since two adjacent bricks share exactly two vertices). Let $N(D)$ denote the number of the H_3-paths needed to remove all separators in $C_{2,3}$. Then, it follows that $N(D) \le N(B_1)+\cdots+N(B_r)+(r-1) \le (2n_1-6)+\cdots+(2n_r-6)+(r-1) = 2\sum_{i=1}^{r} n_i - 6r + (r-1) \le 2n - 6$, where the first inequality follows from the fact that by Lemma 9, the number of H_3-paths added does not exceed the total number of H_3-paths added in all bricks, and at most $(r-1)$ H_3-paths are needed to remove the splitting separators. Furthermore, we have used Lemma 7 in the second inequality. This completes the proof. ■

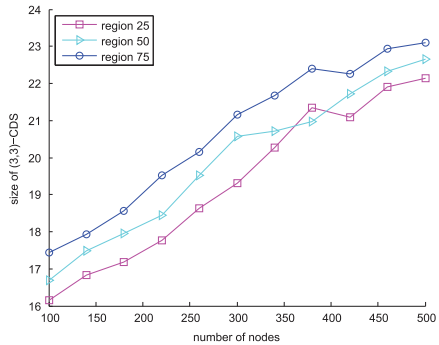


Fig. 11. Size of (3, 3)-CDS computed by SPFA in various virtual space regions.

Theorem 7: Algorithm 2 is a 62.30-approximation for computing (3, 3)-CDS.

Proof: We have an r-approximation algorithm for computing a (2, 3)-CDS with $r = 12.46$ (see [22]), so we can obtain a $C_{2,3}$ such that $|C_{2,3}| \le r|C^{\text{opt}}_{2,3}|$. According to Lemma 10, we have $|C_{3,3}| = |C_{2,3} \cup H| = |C_{2,3}| + |H| \le |C_{2,3}| + 2(2|C_{2,3}| - 6) \le 5r|C^{\text{opt}}_{2,3}| \le 5r|C^{\text{opt}}_{3,3}|$, where $C^{\text{opt}}_{3,3}$ is the optimal solution for the (3, 3)-CDS. This completes the proof. ■

When $m > 3$, first we can compute a $C_{2,m}$ using the existing algorithm in [22]. Then, we augment $C_{2,m}$ to $C_{3,m}$ using our algorithm. Thus, we have the following.

Theorem 8: Algorithm 2 is a $5r$-approximation for computing $(3, m)$-CDS, for $m > 3$, where r is the approximation ratio of the minimum $(2, m)$-CDS in [22].

Theorem 9: The approximation ratio of this algorithm is 170 for $m = 1, 2$.

Proof: First, we prove the conclusion for the case $m = 1$. By the algorithms in [27], we have $C_{1,3} = I_1 \cup I_2 \cup I_3 \cup C$, where I_i is the maximal independent set obtained sequentially and C is some additional vertices added to make I_1 to be connected. Since $|I_i| \le 5|C^{\text{opt}}_{1,1}|$ $(i = 1, 2, 3)$ and $|C| \le 2|I_1|$, we have $|C_{1,3}| \le 17|C^{\text{opt}}_{1,1}|$. Moreover, we know $|C_{2,3}| \le 2|C_{1,3}|$. By our algorithm, $|C_{3,3}| \le 5|C_{2,3}| \le 2 \times 5|C_{1,3}| \le |C^{\text{opt}}_{1,1}|$. Obviously, $|C^{\text{opt}}_{1,1}| \le C^{\text{opt}}_{3,1}$. Thus, $|C_{3,3}| \le 17 \times 2 \times 5|C^{\text{opt}}_{3,1}|$.

For $m = 2$, the approximation ratio can be obtained similarly by $|C^{\text{opt}}_{1,1}| \le |C^{\text{opt}}_{3,2}|$. ■

Theorem 10: Let n be the number of nodes in the original graph $G = (V, E)$. Then, the time complexity of Algorithm 2 is $O(n^3)$.

Proof: Computing a $(2, m)$-CDS takes time $O(n^3)$, and verifying whether $\{u, v\}$ is a separator or not are finished in $O(n^2)$. The time complexity of adding H_3-path is dominated by the Shortest-Path function, which runs in $O(n^2)$. Therefore, the time complexity is at most $O(n^3)$. ■

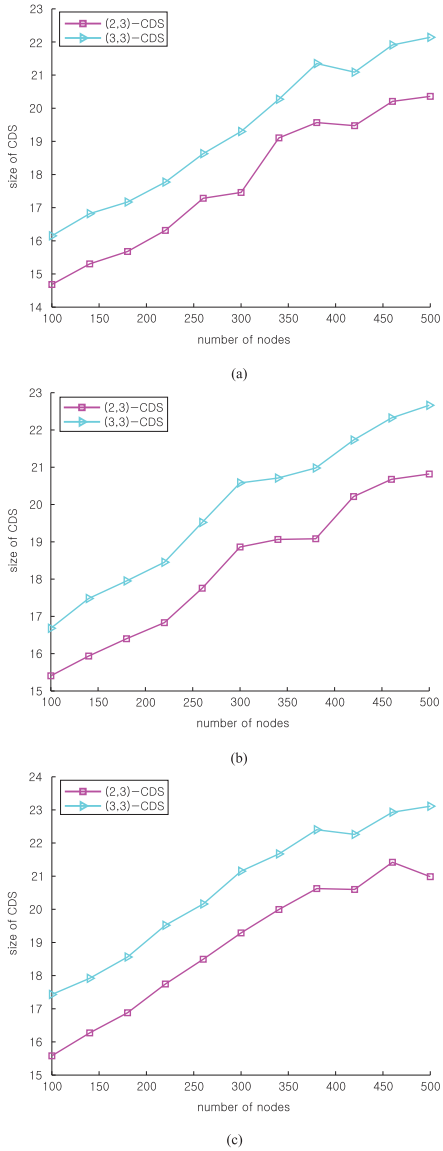


Fig. 12. Comparison between (2, 3)-CDS and (3, 3)-CDS. (a) Comparison in 25×25 region. (b) Comparison in 50×50 region. (c) Comparison in 75×75 region.

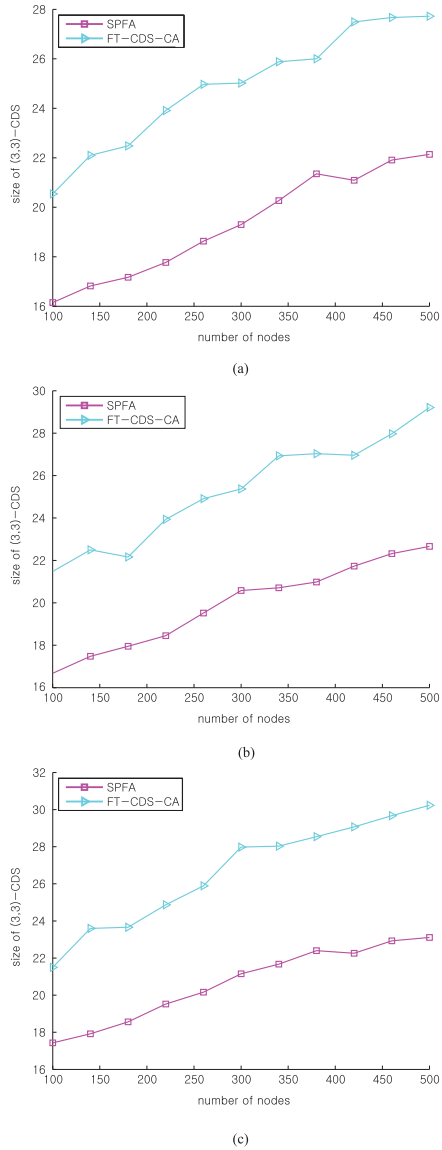


Fig. 13. Comparison with the size of (3, 3)-CDS computation algorithm in [14]. (a) Comparison in 25×25 region. (b) Comparison in 50×50 region. (c) Comparison in 75×75 region.

V. Simulation Results and Analysis

In this section, we conduct simulations to validate the effectiveness of SPFA (Algorithm 2). We randomly generate various network topologies. For each setting, we perform the simulation for 100 times and compute the average value. First, we fix the region size of 25×25, and node size varies from 100 to 500 with the increment of 50. Next, the region size is enlarged up to 50×50 and 75×75, and the node size varies from 100 to 500, so that we can evaluate the SPFA for different node density.

Fig. 11 plots the size of the obtained (3, 3)-CDS computed by SPFA in various virtual space region. From Fig. 11, we observe that with the number of nodes increased, the number of nodes (3, 3)-CDS is increased. The denser the network is, the less nodes (3, 3)-CDS has.

In order to evaluate the performance of SPFA, we first fix the network region, and deploy the nodes randomly. Fig. 12 shows that the size of the (3, 3)-CDS is about 10%–15% larger than (2, 3)-CDS no matter how dense the network is. The number of nodes added to (2, 3)-CDS is increasing along with the number of original network, but relatively slowly. This is because in a smaller region, a large part of (2, 3)-CDS is already connected. However, with the increase of nodes number in our network, the ratio between nodes in (2, 3)-CDS and (3, 3)-CDS is changeless, which indicates that the increase of network size has no significant effect to the performance of our algorithm, and we can expect that our algorithm can perform well in dense networks.

We also evaluate our Algorithm SPFA through comparison with the only existing alternative, FT-CDS-CA in [14]. We set our simulation setting to be same as that of FT-CDS-CA. The method of generating 3-connected graph is same as algorithm FT-CDS-CA. The network size is varied from 100 to 500 nodes with the increment by 50 nodes. For the same network size, we vary side length of region as 25×25, 50×50, 75×75 so that we can test the algorithm for different node density.

Fig. 13 shows that SPFA constructs much smaller (3, 3)-CDS than FT-CDS-CA does in 25×25, 50×50, 75×75 region, respectively. On average, the size of (3, 3)-CDS obtained by SPFA is about 20% smaller than that obtained from FT-CDS-CA. This is because the ratio of SPFA is less than 15%, while the ratio is roughly 20%–30% in the algorithm FT-CDS-CA. From the figures, our algorithm outperforms algorithm FT-CDS-CA with the same size of region. That is because our algorithm always chooses the best node, while FT-CDS-CA may not.

VI. Conclusion

In this paper, we studied the problem of computing $(3, m)$-CDS in wireless networks. We use Tutte decomposition and design a simpler approximation algorithm with much smaller approximation factor for the problem. As our future work, we plan to design constant factor approximation algorithm for the minimum k-connected m-dominating set problem with $k \geq 4$ and $m \geq 1$, which will be a theoretical ground work for constructing a quality virtual backbone with high fault tolerance for wireless networks.

References

[1] W. Wang *et al.*, "A better constant approximation of minimum 3-connected m-dominating set problem in unit disk graph using Tutte decomposition," in *Proc. 34th IEEE INFOCOM*, Hong Kong, Apr. 26–30, 2015, pp. 1796–1804.

[2] R. Tan, G. Xing, B. Liu, J. Wang, and X. Jia, "Exploiting data fusion to improve the coverage of wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 450–462, Apr. 2012.

[3] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. 5th Annu. ACM/IEEE MobiCom*, Washington, DC, USA, Aug. 1999, pp. 152–162.

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

[5] Y. Li, D. Kim, F. Zou, and D.-Z. Du, "Constructing connected dominating sets with bounded diameters in wireless networks," in *Proc. 2nd WASA*, Chicago, IL, USA, Aug. 1–3, 2007, pp. 89–94.

[6] F. Zou, X. Li, D. Kim, and W. Wu, "Construction of minimum connected dominating set in 3-dimensional wireless network," in *Proc. 3rd WASA*, Dallas, TX, USA, Oct. 26–28, 2008, pp. 134–140.

[7] D. Kim, X. Li, F. Zou, Z. Zhang, and W. Wu, "Recyclable connected dominating set for large scale dynamic wireless networks," in *Proc. 3rd WASA*, Dallas, TX, USA, Oct. 26–28, 2008, pp. 560–569.

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

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

[10] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete Math.*, vol. 86, pp. 165–177, Dec. 1990.

[11] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, pp. 374–387, Apr. 1998.

[12] F. Dai and J. Wu, "On constructing k-connected k-dominating set in wireless network," in *Proc. 19th IEEE IPDPS*, 2005, p. 81a.

[13] D. Kim, W. Wang, X. Li, Z. Zhang, and W. Wu, "A new constant factor approximation for computing 3-connected m-dominating sets in homogeneous wireless networks," in *Proc. 29th IEEE INFOCOM*, Mar. 2010, pp. 1–9.

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

[15] W. Shang, F. Yao, P. Wan, and X. Hu, "On minimum m-connected k-dominating set problem in unit disc graphs," *J. Combin. Optimiz.*, vol. 16, pp. 99–106, Dec. 2007.

[16] F. Wang, M. T. Thai, and D.-Z. Du, "2-Connected virtual backbone in wireless network," *IEEE Trans. Wireless Commun.*, vol. 8, no. 3, pp. 1230–1237, Mar. 2009.

[17] M. T. Thai, N. Zhang, R. Tiwari, and X. Xu, "On approximation algorithms of k-connected m-dominating sets in disk graphs," *Theoret. Comput. Sci.*, vol. 358, pp. 49–59, 2007.

[18] N. Zhang, I. Shin, F. Zou, W. Wu, and M. T. Thai, "Trade-off scheme for fault tolerant connected dominating sets on size and diameter," in *Proc. ACM FOWANC*, 2008, pp. 1–8, in conjunction with MobiHoc.

[19] Y. Wu, F. Wang, M. T. Thai, and Y. Li, "Constructing k-connected m-dominating sets in wireless sensor networks," in *Proc. MILCOM*, Orlando, FL, USA, Oct. 29–31, 2007, pp. 1–7.

[20] Y. Wu and Y. Li, "Construction algorithms for k-connected m-dominating sets in wireless sensor networks," in *Proc. 9th ACM MobiHoc*, Hong Kong, May 26–30, 2008, pp. 83–90.

[21] Y. Li, Y. Wu, C. Ai, and R. Beyah, "On the construction of k-connected m-dominating sets in wireless networks," *J. Combin. Optimiz.*, vol. 23, no. 1, pp. 118–139, 2012.

[22] Y. Shi, Y. Zhang, Z. Zhang, and W. Wu, "A greedy algorithm for the minimum 2-connected m-fold dominating set problem," *J. Combin. Optimiz.*, 2014, DOI: 10.1007/s10878-014-9720-6.

[23] W. T. Tutte, *Connectivity in Graphs*. Oxford, U.K.: Oxford Univ. Press, 1966.

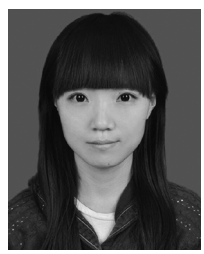
[24] R. Diestel, "Graph theory," in *Graduate Texts in Mathematics*, 3rd ed. Heidelberg, Germany: Springer-Verlag, 2005, vol. 173.

[25] D. B. West, *Introduction to Graph Theory*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[26] G. Chapuy, E. Fusy, M. Kang, and B. Shoilekova, "A complete grammar for decomposing a family of graphs into 3-connected components," *Electron. J. Combinatorics*, vol. 15, no. 1, p. R148, 2008.

[27] Y. Li, S. Zhu, M. T. Thai, and D.-Z. Du, "Localized construction of connected dominating set in wireless networks," in *Proc. NSF TAWN*, Chicago, IL, USA, Jun. 2004, pp. 1–9.

[28] J. E. Hopcroft and R. E. Tarjan, "Dividing a graph into triconnected components," *SIAM J. Comput.*, vol. 2, pp. 135–158, 1973.

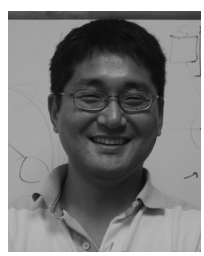


Bei Liu received the M.S. degree in mathematics from Xi'an Jiaotong University, Xi'an, China, and is currently pursuing the Ph.D. degree in mathematics at Xi'an Jiaotong University.

Her major research interests include wireless networking, social networking, and approximation algorithm design and analysis.

Wei Wang received the B.S. degree in applied mathematics from ZheJiang University, Hangzhou, China, in 1991, and the M.S. degree in computational mathematics and Ph.D. degree in mathematics from Xi'an Jiaotong University, Xi'an, China, in 1994 and 2006, respectively.

He is currently a Professor with the School of Mathematics and Statistics, Xi'an Jiaotong University. His research interests include algebraic graph theory and approximation algorithm design and analysis.



Donghyun Kim (S'07–M'10–SM'15) received the B.S. degree in electronic and computer engineering and M.S. degree in computer science and engineering from the Hanyang University, Ansan, Korea, in 2003 and 2005, respectively, and the Ph.D. degree in computer science from the University of Texas at Dallas, Richardson, TX, USA, in 2010.

Currently, he is an Assistant Professor with the Department of Mathematics and Physics, North Carolina Central University, Durham, NC, USA. His main research interest is algorithm design and analysis for various application fields such as security and privacy, Internet

of things, cloud computing, big data, social computing, mobile computing, robotics, cyber-physical systems, and wireless and sensor networking.

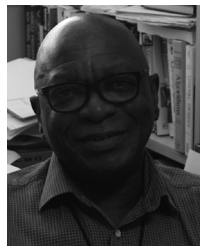
Dr. Kim is a member of the Association for Computing Machinery (ACM). He is an Associate Editor of *Discrete Mathematics, Algorithms and Applications*.

Deying Li received the M.S. degree in mathematics from Huazhong Normal University, Wuhan, China, in 1988, and the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2004.

She is currently a Professor with the Department of Computer Science, Renmin University of China, Beijing, China. Her research includes wireless networks, mobile computing, and algorithm design and analysis.

Jingyi Wang received the B.S. degree in mathematics and applied mathematics from Xi'an Jiaotong University, Xi'an, China, in 2013, and is currently a graduate student in computational mathematics at Xi'an Jiaotong University.

His research interests include algebraic graph theory and approximation algorithm design and analysis.



Alade O. Tokuta (M'86–LM'14) received the Ph.D. degree in electrical engineering and computer science from the University of Florida, Gainesville, FL, USA, in 1984.

Currently, he is a Professor with the Department of Mathematics and Physics, North Carolina Central University, Durham, NC, USA. His research interests include robotics, computer image synthesis/vision, networking, and algorithm design.

Yaolin Jiang received the B. S. degree from Sichuan University, Chengdu, China, in 1985, and the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1987 and 1992, respectively, all in mathematics.

He has been a Professor with the Department of Mathematics, Xi'an Jiaotong University, Xi'an, China, since 1998, and now is also a Changjiang Professor in China. He has published four books and about 200 technical papers in journals and conference proceedings. 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 nonlinear systems, circuit simulation, and parallel processing.