Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5977781, 20 pages
https://doi.org/10.1155/2021/5977781

Research Article

Construction of Quality Virtual Backbones with Link Fault Tolerance in Wireless Sensor Networks

Xinyu Liang[1,2], **Jiarong Liang**[2] **and Weiguang Zhang**[2]

[1]*School of Electrical Engineering, Guangxi University, 530004, China*
[2]*School of Computer, Electronics and Information and Guangxi Key Laboratory of Multimedia Communications and Network Technology, Guangxi University, 530004, China*

Correspondence should be addressed to Jiarong Liang; 13977106752@163.com

Received 27 July 2021; Revised 2 November 2021; Accepted 6 November 2021; Published 8 December 2021

Academic Editor: Ghufran Ahmed

Wireless sensor networks (WSNs) are extensively utilized in various circumstances. For applications, the construction of the virtual backbones (VBs) of WSNs has attracted considerable attention in this field. Generally, a homogeneous WSN is formulated as a unit disk graph (UDG), and the VB of the corresponding WSN is modeled as a connected dominating set (CDS) in the UDG. In certain applications, communication between sensors in a network may fail for various reasons, such as sensor movement, signal interference, and the appearance of obstacles. Consequently, a CDS in a UDG should possess fault tolerance on the edges. In this paper, we introduce a new concept called the 2 edge-connected 2 edge-dominating set ($(2,2)$-ECDS); then, we design an approximation algorithm for computing $(2,2)$-ECDSs in UDGs, the performance ratio of which is 30.51. By means of simulations, we compare our algorithm and existing algorithms in terms of the CDS size, running time, success rate, and network lifetime. The simulation results indicate that our algorithm exhibits better performance and is more suitable for constructing a VB with edge fault tolerance in a WSN.

1. Introduction

Wireless sensor networks (WSNs) are multihop self-organizing network systems that contain numerous sensor nodes. Because no predefined infrastructure is required, WSNs can be flexibly deployed in various circumstances, such as the Internet of Things, ocean monitoring, and traffic surveillance [1–5]. However, the wireless sensors in WSNs are usually powered by batteries, which means that minimizing sensor energy consumption is a critical challenge [6, 7]. Generally, the energy of sensor nodes in a WSN is mainly consumed in the signal transmission process. Specifically, when a signal is transmitted from a source node to a target node, the required energy increases exponentially with increasing distance [8]. Thus, most WSNs use a multihop communication strategy rather than long-range direct communication to enhance the energy efficiency of the sensor nodes [9–14].

In multihop communication, a message from a source node to a target node may be retransmitted many times, which can result in substantial wireless signal interference and collision and waste a large amount of energy. Hence, a high-quality routing path for information transmission is critical. To address this issue in WSN communication, the concept of the VB, inspired by the backbone of wired networks, is supposed to reduce information overload and improve the efficiency and lifespan of the network [1].

In theory, a VB is a subset of sensor nodes in a WSN that meets the following requirements [15]. First, each node in a WSN is adjacent to a node in this subset. Second, any two nodes in the subset have a routing path between them in the VB. To investigate VBs in homogeneous WSNs, the WSN is typically modeled as a UDG $G = (V, E)$, where a vertex in V is a simplification of a sensor node and an edge in E stands for the message transmission between two sensors; a VB of a WSN is modeled as a CDS in the corresponding graph G. In the research of WSNs, since a smaller VB entails less routing overhead and wireless signal collision, the size of the VB is an important index in measuring the quality of the VB. Thus, it is natural to find a minimum CDS (MCDS) of G when constructing a CDS. However, Garey and Johnson [16]

have proven that calculating MCDSs is NP-hard. Consequently, many efforts have been made to obtain an approximate solution to this problem by designing approximation algorithms.

In practical applications, in a WSN, a few sensors and the links between them may be out of operation for various reasons, such as sensor movement, signal interference, and the appearance of obstacles. Thus, the VB for a WSN should be fault tolerant such that it continues to work when some nodes and links fail. Fortunately, in recent years, progress has been made in designing node fault-tolerant VBs for WSNs. In fact, in 2005, Dai and Wu [17] proposed a special CDS, the k-connected k-dominating set, for a node fault-tolerant VB in a WSN against node failure. Later, Thai et al. [18] extended this concept to the famous k-connected m-dominating set ((k, m)-CDS), where $1 \le k \le m$. Since the construction of the minimum $(1, 1)$-CDS is NP-hard, the construction of the minimum (k, m)-CDS is also NP-hard. As a result, numerous researchers have focused on developing approximation algorithms to calculate a (k, m)-CDS with a worst-case performance guarantee for any given pair of integers k and m $(1 \le k \le m)$ [19–27].

Note that the above literature considered only the problem of a fault-tolerant VB against node failure and neglected the impact of edge failure. In practical applications, edge failure may impact the operation of a VB in the considered WSN; in other words, edge failure may lead to failure of the VB even if all nodes in the VB are fault-free. For example, in a WSN with 9 nodes, which is modeled by the UDG shown in Figure 1, the VB consists of nodes u_1, u_3, and u_7. When edge (u_3, u_5) fails, some aspects of the VB, such as domination, will fail, even though each node is fault-free. In practice, edge failure in a WSN may be inevitable due to various factors, such as node movement, signal interference, and obstacles. For example, in the WSN shown in Figure 1, when node u_1 is moved to a position outside of the transmission range of u_3 (see Figure 1(b)), edge (u_1, u_3) fails, which results in subset $\{u_1, u_3, u_7\}$ no longer being a VB.

Note that in a UDG, if a CDS is a node fault-tolerant CDS, such as (k, m)-CDS with $m \ge k \ge 2$, it must be an edge fault-tolerant CDS. Formally, in a UDG $G = (V, E)$, a CDS S is a node fault-tolerant CDS if for any $v \in V$, $S - \{v\}$ is still a CDS in $G - \{v\}$. Similarly, in a UDG $G = (V, E)$, a CDS S is an edge fault-tolerant CDS if for any $e \in E$, S is still a CDS in $G - \{e\}$. Compared to that of constructing an edge fault-tolerant CDS in a UDG, the cost of constructing a node fault-tolerant CDS may be higher. On the other hand, in some UDGs, there may not exist a node fault-tolerant CDS, while there is an edge fault-tolerant CDS. For example, in the UDG G shown in Figure 2, there are no node fault-tolerant CDSs; however, the CDS $\{u_1, u_2, u_3, u_5, u_6\}$ is an edge fault-tolerant CDS because when any edge in G fails, $\{u_1, u_2, u_3, u_5, u_6\}$ is still a CDS. Hence, it is necessary to explore how to construct an edge fault-tolerant CDS in a UDG. Unfortunately, few efforts have been made in this direction in the existing literature.

In this paper, we investigate the construction of an edge fault-tolerant CDS in a UDG. Specifically, this paper introduces a new concept called the 2 edge-connected 2 edge-dominating set (in short, $(2, 2)$-ECDS) and develops a constant factor approximation algorithm to calculate a $(2, 2)$-ECDS in a given UDG.

The main work of our paper is described as follows:

(i) To obtain an edge fault-tolerant VB in a WSN, we introduce a new concept called the $(2, 2)$-ECDS

(ii) A centralized approximation algorithm is designed for the construction of a $(2, 2)$-ECDS in a given UDG, whose performance ratio is proven to be 30.51

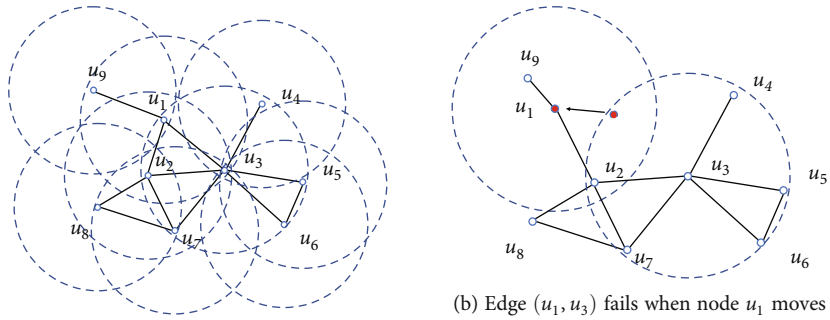
(iii) The designed algorithms are supported by strict theoretical analysis and simulation experiments. In the theoretical analysis, we conduct strict mathematical proof of the correctness of these algorithms. By means of simulation experiments, we illustrate that the CDSs output by our algorithms outperforms those obtained with the state-of-the-art algorithms presented in [25, 26]

The rest of this paper is organized as follows. In Section 2, related research is introduced. In Section 3, an introduction of the definitions, terms, and related results used in the remaining sections is provided. Section 4 presents an approximation algorithm for constructing a $(2, 2)$-ECDS. Section 5 reports simulation results on the comparison between our algorithm and other related algorithms. Finally, in Section 6, we summarize our work and introduce potential directions for future work.

2. Related Work

A WSN can be abstracted as a graph $G = (V, E)$; correspondingly, the VB of the WSN can be abstracted as a CDS of G. In recent years, numerous efforts have been made to construct MCDSs. In a general graph, Guha and Khuller [28] proved that for the MCDS problem, it is impossible to obtain a polynomial time approximation algorithm with a performance ratio $\rho \ln n$ for any $0 < \rho < 1$ unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$), where n is the number of nodes in the graph. They introduced a $2(H(\Delta) + 1)$-approximation algorithm and a $H(\Delta) + 2$-approximation algorithm based on a greedy strategy, where Δ is the maximum degree of nodes in the corresponding network graph and H is the harmonic function. In [29], Du et al. improved this result and presented an $(1 + \varepsilon) \ln (\delta - 1)$-approximation algorithm $(\varepsilon > 0)$, which is the best result in general graphs thus far.

For UDGs, the MCDS problem has been widely investigated, and a variety of results have been obtained [26, 30–39], in which different approximation algorithms have been proposed for this problem. These algorithms usually consist of two phases: (1) a maximal independent set (MIS), which serves as a dominating set in a given UDG, is determined, and (2) some nodes are selected into the MIS such that it becomes connected. Researchers analyzed the performance ratio of the resulting CDS from two aspects: the size of the MIS obtained in the first phase and the number of newly added nodes in the second phase. For the MIS



(a) $\{u_1, u_3, u_7\}$ is not a CDS when edge (u_3, u_5) fails

(b) Edge (u_1, u_3) fails when node u_1 moves outside of the transmission range of u_3, even though all nodes are fault-free

FIGURE 1: An example scenario of a WSN with 9 nodes.

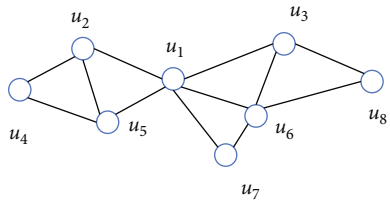


FIGURE 2: An example scenario of a WSN where no $(2, 2)$-CDS exists but a $(2, 2)$-ECDS exists.

size, in [33], Wan et al. demonstrated that the upper bound of an MIS is $4 * \text{opt} + 1$ in a UDG, where opt represents the size of the MCDS in the UDG. Later, this upper bound was improved by [30, 34–37], where [30] obtained an upper bound of $3.399 * \text{opt} + 4.874$, which is the best result thus far. For the size of the newly added nodes, in [33], Wan et al. showed that the size bound generated by their algorithms in the second phase is $|I| - 1$, where $|I|$ is the MIS obtained in the first phase. This result was used by [34, 35] to evaluate the performance ratios of their algorithms. In [31, 38], the Steiner tree strategy was used to design approximation algorithms for the MCDS problem, and the size of the newly added nodes in the second phrase was no more than $(1 + \ln 5)C(T^*)$, where $C(T^*)$ is the number of Steiner nodes in the minimum Steiner tree containing the MIS obtained in the first phase. To our knowledge, this is the best result thus far.

In research on the construction of node fault-tolerant VBs, Dai and Wu [17] introduced a concept called the (k, k)-CDS. For a (k, k)-CDS in a given $G = (V, E)$, after $(k - 1)$ nodes are deleted from G, the set of residual nodes in the (k, k)-CDS is still a CDS for the subgraph induced by the residual nodes of G. They presented three algorithms to compute a (k, k)-CDS for a given UDG. Unfortunately, they did not present the performance ratios of these three algorithms. In [18], Thai et al. extended this concept to the famous k-connected m-dominating set, where $1 \le k \le m$. In their paper, a centralized approximation algorithm was provided for the minimum (k, m)-CDS problem. Since then, numerous constant factor approximation algorithms for the construction of (k, m)-CDSs have been proposed. In [40], Wu et al. presented a distributed algorithm with a performance ratio of $5/m(k^2 + 1)(m + 42)$. In [25], Shang et al. proposed another approximation algorithm, whose performance ratio is $(5 + 25/m)(2 \le m \le 5)$ or 11 $(m > 5)$, for constructing a $(2, m)$-CDS. In [26], Shi et al. noted mistakes in the proof process of the performance ratio of the algorithm in [25] and found that the ratio should be $(15 + 15/m)$ $(2 \le m \le 5)$ or 21 $(m > 5)$ rather than $(5 + 25/m)$ $(2 \le m \le 5)$ or 11 $(m > 5)$. Moreover, Shi et al. [26] used a submodular function and greedy strategy to design a new approximation algorithm with a performance ratio of $\alpha + 2$ $(1 + \ln \alpha)$, where α is the performance ratio of the algorithm for constructing the $(1, m)$-CDS. In 2010, Kim et al. [23] introduced a $15r$- approximation algorithm for computing a $(3, m)$-CDS in a UDG, where r is the performance ratio of the algorithm for the $(2, m)$-CDS $(m \ge 2)$ problem. In [41], Liu et al. used Tutte decomposition to design a $5r$-approximation algorithm to calculate a $(3, m)$-CDS $(m \ge 3)$ in a UDG, where r is the performance ratio of the algorithms for the $(2, m)$-CDS $(m \ge 2)$ problem. When $m = 3$, they noted that the performance ratio of their algorithm is 62.30, which is better than the 580/3 performance ratio achieved by [23]. In [8], Wang et al. introduced the concepts of "*good points*" and "*bad points*" and proposed an algorithm to calculate a (k, m)-CDS in a given UDG, where $k = 3, 4$ and $m \ge k$. The main idea of their algorithm is to first obtain a $(k - 1, m)$-CDS by means of an existing algorithm and then convert all degree-$(k - 1)$ bad points in the $(k - 1, m)$-CDS into good points by adding "H-paths." Second, they construct a simplified auxiliary graph for the CDS obtained in each iteration. Finally, the (k, m)-CDS is constructed by means of the simplified auxiliary graph. They achieved a performance ratio of $5\beta_2$ for the $(3, m)$-CDS problem ($7\beta_3$ for the $(4, m)$-CDS problem), where β_k is the performance ratio of the algorithm proposed in [26] for (k, m)-CDS problem $(k = 2, 3)$.

In recent years, research on link failure in WSN communication has attracted considerable attention. In [42], Zonouz et al. proposed a model, called time-dependent log-normal shadowing radio propagation, to analyze and estimate the reliability of a wireless link under consideration of the power consumption in different modes and wireless channel conditions. In [43], Fu et al. noted that the capacity of links is significantly limited in most WSNs. They believed

that when the load of a few links in the network exceeds their capacity, link failures may occur, which may cause paralysis of the global network. Later, in [44], Draz et al. provided a link failure detection algorithm to proactively diagnose faulty links in a WSN and introduced a recovery mechanism against link failures. In [45], Xu et al. used independent Bernoulli processes to describe link failures in a WSN and employed a Kalman-consensus filter and CI algorithm to estimate these link failures through a hierarchical fusion method. In [46], to improve the robustness of VBs, Liang et al. proposed a new concept, robust VB, and corresponding algorithms against link failures in WSNs with unstable transmission ranges. In their work, a more robust VB, named the d-robust CDS, is obtained by their algorithms, and based on this VB, the communication in the corresponding WSN does not fail when $R_e \geq (1-d) * R$, where R_e is the effective transmission range of the nodes and R is the initial transmission range of the nodes.

As described above, link failures can impact WSN communications. A natural idea is to investigate VBs with link fault tolerance, namely, the problem of edge fault-tolerant VBs against link failure. Unfortunately, research on the construction of edge fault-tolerant VBs in WSNs remains scarce. In this paper, we consider this problem and introduce the concept of the $(2,2)$-ECDS for UDG, which is a kind of edge fault-tolerant VB in the corresponding homogeneous WSN, and propose an approximation algorithm for the construction problem of the $(2,2)$-ECDS.

3. Preliminaries

For convenience, some concepts, notations, and knowledge related to graph theory are now introduced. In this paper, all considered graphs are connected unless otherwise specified.

3.1. System Model and Problem Definition. In this paper, we adopt the following assumptions: the sensors in the considered WSN are randomly placed in a given area and immobile thereafter; each sensor has the same transmission radius and initial energy; two sensors can communicate with each other if and only if one of them is within the transmission range of the other; and in the considered WSN, each link between sensors is bidirectional. We abstract each sensor of the WSN as a node of a graph and each communication link between two sensors as an edge between the corresponding nodes. Thus, a WSN is modeled as an undirected connected graph $G = (V, E)$.

The problem that we address in this paper is called the *minimum* $(2,2)$ *-ECDS problem*. For the sake of understanding the minimum $(2,2)$-ECDS problem, we first introduce some necessary concepts related to this problem.

Definition 1 (2 edge-connected graph). A graph $G = (V, E)$ is called a 2 edge-connected graph if for any $e \in E$, $G\langle E - \{e\}\rangle$ is connected, where $G\langle E - \{e\}\rangle$ is the spanning subgraph induced by edge set $E - \{e\}$. In particular, a graph with a single node is defined as a 2 edge-connected graph.

Definition 2 (2 edge-dominating set). In a given graph $G = (V, E)$, a subset $D \subseteq V$ is called a 2 edge-dominating set if for each node $v \in V - D$, $|E(D, v)| \geq 2$.

In a given UDG $G = (V, E)$, let $u \in V$ and $D \subseteq V$. If $|E(D, u)| \geq 2$, then we say that u is 2 edge-dominated by D.

Definition 3 (2 edge-connected 2 edge-dominating set). In a given UDG $G = (V, E)$, a CDS $D \subseteq V$ is a $(2,2)$-ECDS if the following conditions hold:

(1) D is a 2 edge-dominating set of G

(2) $G[D]$ is a 2 edge-connected graph

A $(2,2)$-ECDS with the minimum cardinality is called a minimum $(2,2)$-ECDS.

Definition 4 (minimum $(2,2)$-ECDS problem). For a given graph $G = (V, E)$, the minimum $(2,2)$-ECDS problem is the problem of constructing a minimum $(2,2)$-ECDS for G.

Note that because the MCDS problem for a graph is NP-hard, the minimum $(2,2)$-ECDS problem for a graph is also NP-hard. In this paper, we will design an approximation algorithm for the minimum $(2,2)$-ECDS problem.

3.2. Notations. In this subsection, we use $G = (V, E)$ to denote a UDG.

(i) $G\langle E^*\rangle$: the spanning subgraph induced by $E^* \subseteq E$, namely, $G\langle E^*\rangle = (V, E^*)$

(ii) $G[S]$: the subgraph induced by $S \subseteq V$

(iii) $E(v)$: the edge subset, where each edge is incident to $v \in V$, namely, $E(v) = \{e \in E | v \text{ is one of } e\text{'s endpoints}\}$

(iv) $E(S, v)$: $E(S, v) = \{e \in E | \text{one endpoint of } e \text{ is } v \text{ and another endpoint of } e \text{ is in } S\}$, where $v \in V - S$

(v) $N(u)$: the set of neighbor nodes of $u \in V$

(vi) $N[u]$: $N[u] = N(u) \cup \{u\}$

(vii) ID_u: the ID of node u

(viii) e_u: the energy of node u

(ix) $E(G)$: the set consisting of all edges of G

(x) $V(G)$: the set consisting of all nodes of G

3.3. Definitions

Definition 5 (weight function $W(x_1, x_2, x_3)$). Given two 3-tuple variables (x_{1i}, x_{2i}, x_{3i}) and (x_{1j}, x_{2j}, x_{3j}), for a weight function $W : (x_1, x_2, x_3) \mapsto R$, $W(x_{1i}, x_{2i}, x_{3i}) > W(x_{1j}, x_{2j}, x_{3j})$ if and only if at least one of the following conditions is true:

(i) $x_{1i} > x_{1j}$

(ii) $x_{1i} = x_{1j}$ and $x_{2i} > x_{2j}$

(iii) $x_{1i} = x_{1j}$ and $x_{2i} = x_{2j}$ and $x_{3i} < x_{3j}$

Definition 6 (bridge). In a given UDG $G = (V, E)$, $e^* \in E$ is called a bridge of G if $G\langle E - \{e^*\}\rangle$ is disconnected.

According to Definitions 1 and 6, a given UDG $G = (V, E)$ is a 2 edge-connected graph if and only if each edge $e \in E$ is not a bridge of G.

Definition 7 (edge-disjoint). In a given UDG $G = (V, E)$, let P_1 and P_2 be two different paths. Then, P_1 and P_2 are called edge-disjoint if $E(P_1) \cap E(P_2) = \varnothing$.

Definition 8 (simple cycle). In a given UDG $G = (V, E)$, let $P = x_1 \leftrightarrow x_2 \leftrightarrow \cdots \leftrightarrow x_k$ be a path of G. The path P is called a simple cycle in G if $x_1 = x_k$ and $x_i \neq x_j (i, j = 2, 3, \cdots, k-1, i \neq j)$.

According to Whitney's theorem [47], the proposition that for a given $G = (V, E)$, e^* is a bridge of G is equal to any one of the following propositions.

(1) In G, $\exists x, y \in V$ such that all paths between x and y contain edge e^*

(2) G has no simple cycle containing e^*

Definition 9 (2E-block). In a given UDG $G = (V, E)$, let $S \subseteq V$. $G[S]$ is a 2E-block of G if one of the following conditions holds:

(1) $S \subset V$ consists of a single node, and for any nonempty subset $S' \subseteq V - S$, $G[S \cup S']$ is not 2 edge-connected

(2) $S \subset V$ consists of at least two nodes; $G[S]$ is a 2 edge-connected graph; and for any nonempty subset $S' \subseteq V - S$, $G[S \cup S']$ is not 2 edge-connected

(3) $S = V$ and G is a 2 edge-connected graph

Remark 10. According to the definitions of the 2E-block and 2 edge-connected graphs, for a given graph UDG $G = (V, E)$ and a set $S \subseteq V$, we have the following conclusions:

(1) If $G[S]$ is a 2E-block of G, then $G[S]$ is 2 edge-connected blocks

(2) $G[S]$ is the unique 2E-block of $G[S]$ if and only if $G[S]$ is 2 edge-connected

Definition 11 ($L_k(S)$path). In a given UDG $G = (V, E)$, let $S \subseteq V$ and $x, y \in S$. A path from x to y with $k + 2$ nodes is called a $L_k(S)$ path if its k inner nodes belong to $V - S$.

Lemma 12. *In a given UDG $G = (V, E)$, for any $u, v \in V$, there exist two paths that are edge-disjoint between u and v if and only if G is a 2 edge-connected graph.*

According to Definition 7 and Whitney's theorem, the proof of Lemma 12 is easy to obtain.

4. Algorithm and Analysis

In this section, we present an algorithm (called LLZ20) to construct a $(2, 2)$-ECDS in a 2 edge-connected UDG G. LLZ20 includes three main parts. First, a CDS in G is constructed with the two-stage approximation algorithm (called ACC11) in [32]. Next, a connected 2 edge-dominating set is calculated by adding some nodes to the CDS obtained in the first part; this can be implemented by calling Algorithm 2, whose flow chart is shown in Figure 3. Finally, $L_k(S)$ paths are added to connect the nodes of the connected 2 edge-dominating set obtained in the second part to make it a $(2, 2)$-ECDS; this can be implemented by calling Algorithm 3, whose flow chart is shown in Figure 4. In Algorithm 3, to compute 2E-blocks in a subgraph, we need to call another algorithm, Algorithm 4, whose flow chart is shown in Figure 5. LLZ20 is described formally in Algorithm 1.

For readers to understand LLZ20 more intuitively, an example network graph is presented in Figure 6 to illustrate the steps of LLZ20. For convenience, let the initial energy of each node be the same.

(1) For a given WSN, which is abstracted as the network graph $G = (V, E)$ (shown in Figure 6(a)), LLZ20 calls ACC11 to construct a CDS in G (line 1). In this process, u_1, u_2, and u_4 are selected consecutively to form a CDS, and then, pseudocontrol node u_4 is deleted because u_4 does not affect the connectivity of the CDS. When this step is complete, $\{u_1, u_2\}$ (blue nodes shown in Figure 6(b)) is the resulting CDS S_{CDS}

(2) In line 2 of LLZ20, Algorithm 2 is called to calculate a dominating set D (green nodes shown in Figure 6(c)) in $G[V - \{u_1, u_2\}]$. Before Algorithm 2 is executed, nodes u_3, u_4, u_7, u_8, and u_9 are not 2 edge-dominated by the CDS $\{u_1, u_2\}$. When Algorithm 2 is executed, u_4 and u_3 are consecutively selected to obtain a connected 2 edge-dominating set $S_0 = \{u_1, u_2, u_3, u_4\}$, and the pseudocontrol node set VD is an empty set (see Figure 6(c) for details)

(3) In line 3 of LLZ20, Algorithm 3 is called to calculate the set connector, which consists of the inner nodes of the selected $L_k(S)$ path. Specifically, before Algorithm 2 is executed, $\{u_1, u_2, u_3, u_4\}$ is not 2 edge-connected. When Algorithm 2 is executed, the path $u_1 \leftrightarrow u_5 \leftrightarrow u_2$ and the path $u_2 \leftrightarrow u_7 \leftrightarrow u_4$ are consecutively selected; in other words, u_5 and u_7 are consecutively selected into the set connector. After this step is finished, $S_1' = \{u_1, u_2, u_3, u_4, u_5, u_7\}$ is the resulting $(2, 2)$-ECDS (see Figure 6(d) for details)

Input: A 2 edge-connected UDG $G = (V, E)$.
Output: A $(2, 2)$-ECDS S_1'.
1 Construct a CDS S_{CDS} with ACC11 in G.
2 Construct a connected 2 edge-dominating set S_0 and a pseudocontrol node set VD with Algorithm 2 based on S_{CDS}.
3 Construct a $(2, 2)$-ECDS S_1' with Algorithm 3 based on S_0 and VD.
4 **return** S_1'.

Algorithm 1: LLZ20.

Input: A connected graph $G = (V, E)$ and a CDS S_{CDS} of G.
Output: A connected 2 edge-dominating set S_0 and a pseudocontrol node set VD.
1 $B_0 = V - S_{CDS}$
2 **for** each node $v \in B_0 k$**do**
3 **If**$|E(S_{CDS}, v)| \geq 2$**then**
4 $B_0 = B_0 - \{v\}$
5 **end if**
6 **end for**
7 $D = \emptyset$, $B = B_0$, $L_1 = V - S_{CDS} - B_0$
8 **while** $B \neq \emptyset$ **do**
9 Find a node $u \in B$ with the highest $W(e_u, |N(u) \cap B|, ID(u))$.
10 **if**$|N(u) \cap B| = 0$**then**
11 $VD = VD \cup \{u\}$
12 **end if**
13 $D = D \cup \{u\}$ and $B = B - N[u]$.
14 **end while**
15 $S_0 = S_{CDS} \cup D$
16 **return** S_0 and VD.

Algorithm 2: Calculate a 2 edge-dominating set.

4.1. Analysis of Algorithm Correctness. In this subsection, we provide proof of the correctness of our algorithms and show that a $(2, 2)$-ECDS must be obtained after executing LLZ20. For this purpose, we need to show only that the following three results are true: (1) ACC11 in [32] can generate a CDS, (2) Algorithm 2 can generate a connected 2 edge-dominating set, and (3) Algorithm 3 can generate a $(2, 2)$-ECDS.

According to [32], the first result is true. Next, we show the correctness of the second result that Algorithm 2 can generate a connected 2 edge-dominating set.

Lemma 13. *After executing Algorithm 2, the output S_0 is a connected 2 edge-dominating set.*

Proof. We need to prove that $G[S_0]$ is connected and that S_0 is a 2 edge-dominating set. First, since S_{CDS} is a CDS and $S_0 = S_{\mathrm{CDS}} \cup D$, $G[S_0]$ is connected. Next, we prove that S_0 is a 2 edge-dominating set. Let $L_2 = B_0 - D$, where B_0 (D, respectively) is B_0 (D, respectively) obtained after executing line 15. Then, $V = S_0 \cup L_1 \cup L_2$. Let $v \in V - S_0$; then, $v \in L_1 \cup L_2$. We claim that there are two edges with a common endpoint v such that their other endpoints are in S_0. If $v \in L_1$, then according to lines 2-7, $|E(S_{\mathrm{CDS}}, v)| \geq 2$, our claim is true. If $v \in L_2$, then according to lines 8-14, after executing line 14, D is an MIS in $G[B_0]$, which means there is an edge with one endpoint v and another endpoint belonging to D. On the other hand, since S_{CDS} is a CDS, there is an edge with endpoint v and another endpoint belonging to S_{CDS}. Hence, our claim is true. According to the above analysis, S_0 is a 2 edge-dominating set. □

According to the above lemma and lines 10-12 of Algorithm 2, it is easy to obtain the following corollary.

Corollary 14. *After Algorithm 2 is executed, for any $S^* \subseteq VD$, $S_0 - S^*$ is a 2 edge-dominating set in $G[V - VD]$.*

Now, we show that S_1' (the result of Algorithm 3) is a $(2, 2)$-ECDS. Before we present the proof, let us first introduce some definitions and lemmas used in the following discussions. The following preliminaries, which serve as proof of the correctness of the results, are necessary.

For node u chosen in line 3, let $E_u = \{e_1, e_2, \cdots, e_{k_u}\}$ (line 9), S_j^u be a set of red nodes with the same Eblock_ID(u) after executing the jth iteration of the *for*-loop (lines 10-24) in Algorithm 4 ($j = 1, 2, \cdots, k_u$), which is the iteration for e_j (line 10). Let k_j^u be the number of Eblock_IDs of the red nodes in C (line 11) before executing the jth iteration of the *for*-loop (lines 10-24), and let SP_j be the subset of S_j^u that consists of all nonred nodes in C (line 11) before executing the jth

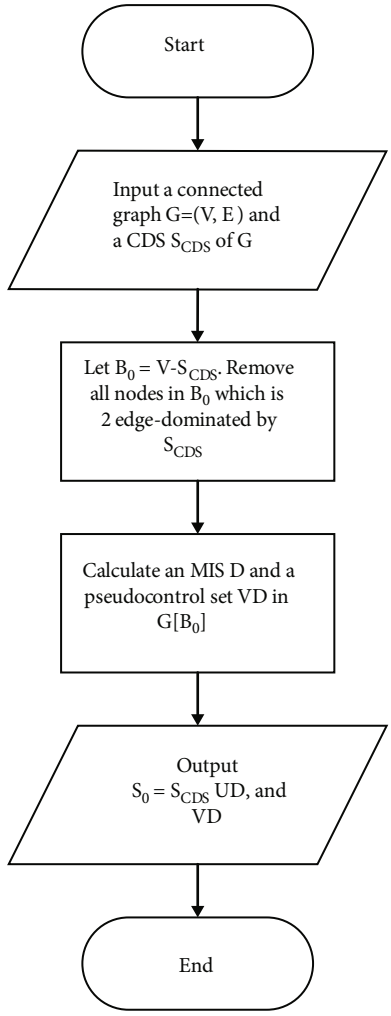


Figure 3: The flow chart of Algorithm 2.

iteration of the *for*-loop (lines 10-24) in Algorithm 4. When $k_j^u \geq 1$, let $S_j(i)(i = 1, 2, \cdots, k_j^u)$ be the nonempty subset of S_j^u, which consists of all red nodes with the same *E*block_ID before executing the *j*th iteration of the *for*-loop (lines 10-24) in Algorithm 4. Then, $S_j^u = S_j(1) \cup S_j(2) \cup \cdots \cup S_j(k_j^u) \cup SP_j$. In this case, $\{S_j(1), S_j(2), \cdots, S_j(k_j^u), SP_j\}$ is called the *TC decomposition* of S_j^u, $S_j(i)$ is called a *red TC decomposition piece* of S_j^u, and SP_j is called a *nonred TC decomposition piece* of S_j^u. Clearly, if there is no TC decomposition in S_j^u, then $S_j^u = SP_j$.

For a better understanding of TC decomposition, let us consider the network graph G_1 shown in Figure 7, which is one part of the network obtained before executing the *j*th iteration of the *for*-loop (lines 10-24) in Algorithm 4, where the integer j is determined as follows. Assume that in network graph G_1, u_{10} is the node u chosen by line 3 in Algorithm 4 and $E_{u_{10}} = \{(u_{10}, u_2), (u_{10}, u_4)\}$, the *j*th iteration of the *for*-loop (lines 10-24) is referred to as the iteration, in which the edge $(u_{10}, u_2) \notin$ BRANCH is considered (line 10). Assume that $C = u_2 \leftrightarrow u_3 \leftrightarrow u_4 \leftrightarrow u_5 \leftrightarrow u_9 \leftrightarrow u_{10} \leftrightarrow u_2$ is the simple circle determined by line 11 in the *j*th iteration, where $E\text{block\_ID}(u_2) = E\text{block\_ID}(u_3) = E\text{block\_ID}(u_4)$ and $E\text{block\_ID}(u_2) \neq E\text{block\_ID}(u_5)$ before the *j*th iteration of the *for*-loop (lines 10-24). Then, $S_j^{u_{10}} = \{u_1, u_2, \cdots, u_{10}\}$, $k_j^{u_{10}} = 2$, $S_1(1) = \{u_1, u_2, u_3, u_4\}$, $S_1(2) = \{u_5, u_6, u_7, u_8\}$, $SP_1 = \{u_9, u_{10}\}$. Finally, $\{S_1(1), S_1(2), SP_1\}$ is a TC decomposition of $S_j^{u_{10}}$.

Lemma 15. *If there is no TC decomposition in S_j^u, then $G[S_j^u]$ contains a simple circle with all nodes of S_j^u.*

Proof. According to the above discussion, it is easy to see that the result is true. □

Lemma 16. *Suppose that $k_j^u \geq 1$ and $\{S_j(1), S_j(2), \cdots, S_j(k_j^u), SP_j\}$ is the TC decomposition of S_j^u, where $j = 1, 2, \cdots, k_u$. If $G[S_j^{(i)}](i = 1, 2, \cdots, k_j^u)$ is 2 edge-connected, then $G[S_j^u]$ is 2 edge-connected.*

Proof. According to the definition of SP_j and line 19, $SP_j \subseteq F(C)$, where $F(C)$ is the node set of the simple circle C selected in line 11 of Algorithm 4. Clearly, $S_j(1) \cup S_j(2) \cup \cdots \cup S_j(k_j^u) \cup F(C) = S_j^u$. We claim that $S_j(i) \cap F(C) \neq \varnothing$ $(1 \leq i \leq k_j^u)$. Otherwise, after executing the *j*th iteration of the *for*-loop (lines 10-24) in Algorithm 4, the *E*block_IDs of the nodes in $S_j(i)$ are not equal to $E\text{block\_ID}(u)$, which implies that $S_j(i)$ is not contained in S_j^u, a contradiction. Let $E_1, E_2, \cdots, E_{k_j^u}$ and $E_{k_j^u+1}$ be the edge sets in $G[S_j(1)]$, $G[S_j(2)], \cdots, G[S_j(k_j^u)]$ and C, respectively. Construct a new graph $G^* = (V^*, E^*)$, where $V^* = S_j(1) \cup S_j(2) \cup \cdots \cup S_j(k_j^u) \cup F(C)$, $E^* = E_1 \cup E_2 \cup \cdots \cup E_{k_j^u} \cup E_{k_j^u+1}$. Clearly, G^* is connected: we claim that G^* is 2 edge-connected. We need to prove only that for any edge $e \in E^*$, $G^*\langle E^* - \{e\}\rangle$ is connected. □

Case 1. $e \in E_i(i = 1, 2, \cdots, k_j^u)$. Since $G[S_j(i)]$ is 2 edge-connected, $G[S_j(i)] - \{e\}$ is still connected. According to $S_j(i) \cap F(C) \neq \varnothing$ and the fact that C is connected, we have that $G^*\langle E^* - \{e\}\rangle$ is connected.

Case 2. $e \in E_{k_j^u+1}$. Since C is a simple circle, $C - \{e\}$ is still connected. Note that $S_j(i) \cap F(C) \neq \varnothing$ and $G[S_j(i)]$ are 2 edge-connected $(i = 1, 2, \cdots, k_j^u)$; thus, we have that $G^*\langle E^* - \{e\}\rangle$ is connected.

Hence, G^* is 2 edge-connected. Since G^* is a spanning subgraph of $G[S_j^u]$, $G[S_j^u]$ is 2 edge-connected.

Lemma 17. *After executing line 29 in Algorithm 4, $G[R_i](i = 1, 2, \cdots, M)$ is a 2E-block in $G[S]$.*

Proof. According to line 6, lines 14-15, and lines 18-22, R_i $(i = 1, 2, \cdots, M)$ consists of red nodes with the same *E*block_ID or a single blue node. Consider the following cases. □

Input: A 2 edge-connected graph $G = (V, E)$, a connected 2 edge-dominating set S_0 in G and the corresponding pseudocontrol node set VD.
Output: A $(2, 2)$-ECDS S_1' of G.
1 Color all nodes white and assign each node $x \in V$ a *super* ID (called Eblock ID, denoted by $Eblock\_ID(x)$). Initially, let $Eblock\_ID(x) = \infty$ for each node $x \in V$.
2 Let $connector = \varnothing$, and calculate all 2E-blocks with Algorithm 4 in $G[S_0 \cup connector]$.
3 **while** the number of 2E-blocks in $G[S_0 \cup connector]$ is more than 1 **do**.
4 **if** there exists an $L_1(S_0 \cup connector)$ path connecting at least two 2E-blocks, **then**
5 Find an $L_1(S_0 \cup connector)$ path L_1^* whose inner node x connects the most 2E-blocks.
6 $connector = connector \cup \{x\}$. Recompute the 2E-blocks in $G[S_0 \cup connector]$ with Algorithm 4.
7 **else**
8 Find an $L_2(S_0 \cup connector)$ path L_2^* such that the number of 2E-blocks connected by its inner nodes is the most.
9 $connector = connector \cup \{y, z\}$, where y and z are the inner nodes of L_2^*. Recompute the 2E-blocks in $G[S_0 \cup connector]$ with Algorithm 4.
10 **end if**
11 **end while**
12 $S_1 = S_0 \cup connector$ and $S_1' = S_1$
13 **for** each node s in VD**do**
14 **if** $G[S_1' - \{s\}]$ is 2-edge connected **then**
15 $S_1' = S_1' - \{s\}$
16 **end if**
17 **end for**
18 **return**S_1'

Algorithm 3: Calculate a 2 edge-connected set.

Case 1. R_i consists of a single blue node. Let x be this node. Clearly, x is not in any simple circle in $G[S_j^u]$ (otherwise, according to lines 10-24, x would be colored red). We claim that for any nonempty subset $S' \subseteq S - \{x\}$, $G[S' \cup \{x\}]$ is not 2 edge-connected. In contrast, assume that there exists a nonempty subset $S' \subseteq S - \{x\}$ such that $G[S' \cup \{x\}]$ is 2 edge-connected. Let $y \in S'$; then, there exist two edge-disjoint paths between x and y in $G[S' \cup \{x\}]$ that form a simple circle and contain node x, a contradiction. Thus, this claim is true. According to the definition of a $2E$-block, $G[R_i]$ is a $2E$-block.

Case 2. R_i consists of red nodes with the same Eblock_ID.

According to the definition of R_i (line 29) and the structure of the *for*-loop (lines 2-28), we conclude that there is some selected node $u \in S$ (line 3) with $E(S, u) - \text{BRANCH} \neq \phi$ and some integer $j(1 \leq j \leq k_u)$ such that the corresponding S_j^u is R_i. Now, consider the following two subcases:

Case 1. There is no TC decomposition in R_i (S_j^u).

According to Lemma 15, $G[R_i]$ contains a simple circle with all nodes in R_i, which implies that $G[R_i]$ is 2 edge-connected nodes. Next, we show that for any nonempty set $S' \subseteq S - R_i$, $G[S' \cup R_i]$ is not 2 edge-connected. To the contrary, assume that $G[S' \cup R_i]$ is 2 edge-connected. Let $x \in S'$ and $y \in R_i$; then, there are two edge-disjoint paths between x and y in $G[S' \cup R_i]$ that form a simple circle $C(x, y)$. $C(x, y)$ has at least one edge that does not belong to BRANCH. Note that for any $(x_1, y_1) \notin$ BRANCH, there exists a unique simple cycle containing (x_1, y_1), whose other edges belong to BRANCH. Hence, there exists a simple cycle $\bar{C}(x, y)$ containing x, y and an edge $(x_2, y_2) \notin$ BRANCH, whose other edges belong to BRANCH. Without loss of generality, suppose that $ID_{x_2} < ID_{y_2}$. When x_2 is selected in line 3, $(x_2, y_2) \in E_{x_2}$. Then, a simple circle C (line 11), whose edges are all in set BRANCH $\cup \{(x_2, y_2)\}$, is determined in the iteration in which (x_2, y_2) is chosen (line 10). Clearly, C is $\bar{C}(x, y)$. After this iteration is finished, x, y are colored red, and Eblock_ID$(x) = E$block_ID(y). Furthermore, after executing line 9, it is still true that Eblock_ID$(x) = E$block_ID(y), which implies that $x, y \in R_i$, a contradiction to the assumption that $x \in S'$. Hence, $G[S' \cup R_i]$ is not 2 edge-connected. As a result, $G[R_i]$ is a $2E$-block.

Case 2. There is a TC decomposition in $R_i(S_j^u)$.

We first prove that $G[R_i]$ is 2 edge-connected. Assuming that $G[R_i]$ is not a 2 edge-connected graph, we will arrive at a contradiction. Let $\{S_j(1), S_j(2), \cdots, S_j(k_u), SP_j\}$ be the TC decomposition of R_i; then, according to Lemma 16, there must exist $t_1(t_1 \in \{1, 2, \cdots, k_u\})$ such that $G[S_j(t_1)]$ is not 2 edge-connected. According to the structure of the *for the*-loop (lines 2-28), there is some $u_1 \in S$ and some integer j_1 such that $S_j(t_1) = S_{j_1}^{(u_1)}$.

If there is no TC decomposition in $S_j(t_1)$, then according to Lemma 15, $G[S_j(t_1)]$ is 2 edge-connected, a contradiction. If there is a TC decomposition in $S_j(t_1)$, we have that there exists a $S_{j_1}(t_2) \subseteq S_j(t_1)$ such that $G[S_{j_1}(t_2)]$ is not 2 edge-connected, where $t_2 \in \{1, 2, \cdots, k_{u_1}\}$. Repeating this procedure,

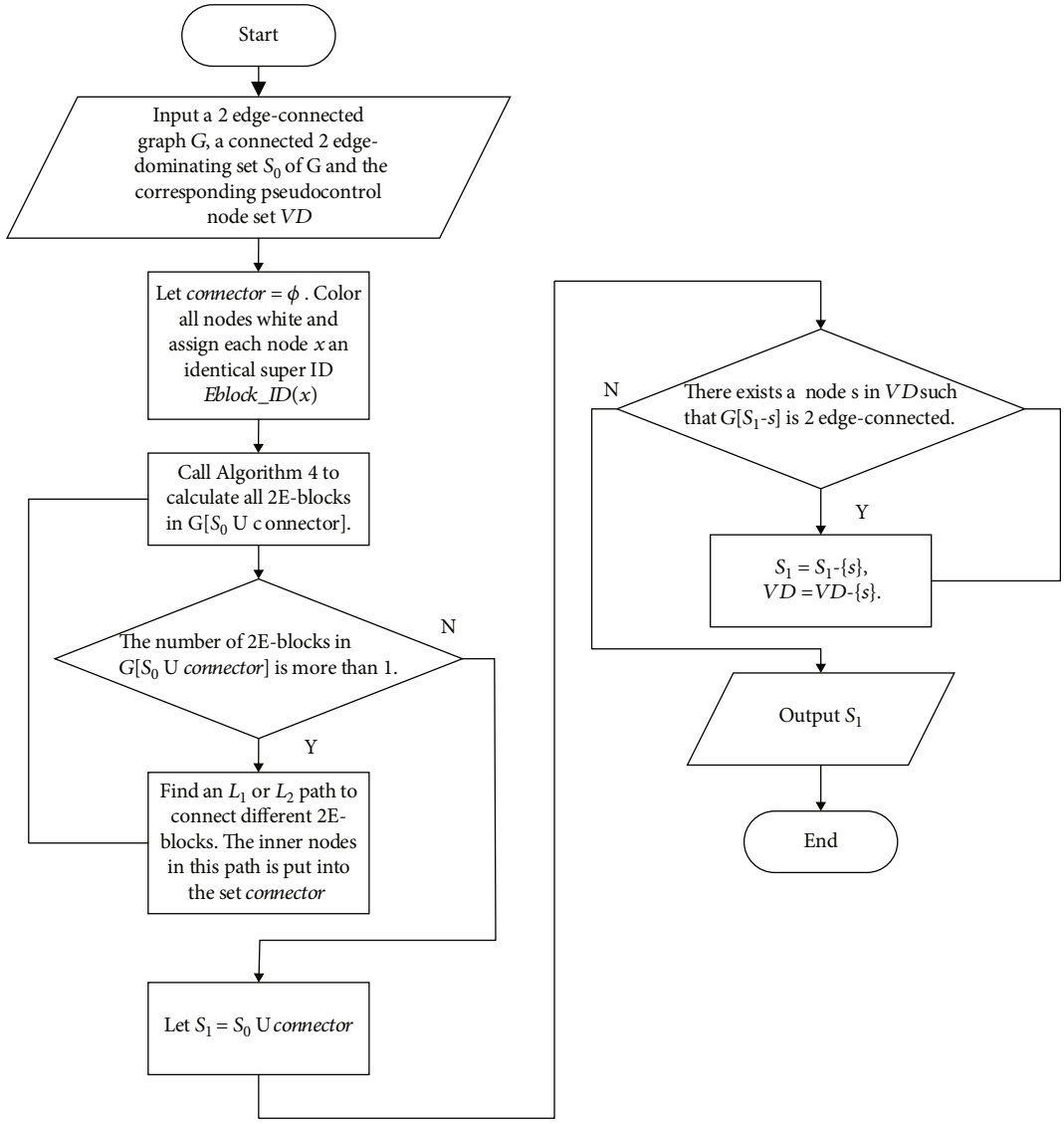


Figure 4: The flow chart of Algorithm 3.

we obtain a series of sets $S_{j_0}(t_1), S_{j_1}(t_2), \cdots, S_{j_{m-1}}(t_m)$ and $S_{j_m}(t_{m+1})$ that satisfy the following conditions:

(1) $S_{j_i}(t_{i+1})$ is a red TC decomposition piece of $S_{j_{i-1}}(t_i)$, $1 \le i \le m$

(2) $G[S_{j_i}(t_{i+1})]$ $(0 \le i \le m)$ is not 2 edge-connected

(3) $S_{j_i}(t_{i+1})$ $(0 \le i \le m-1)$ has a TC decomposition, and $S_{j_m}(t_{m+1})$ has no TC decomposition

Here, $S_{j_0}(t_1) = S_j(t_1)$. Notably, the iteration of this procedure can terminate because each S_j^u that has a TC decomposition is determined by an edge in $G[S] - \text{BRANCH}$ and the number of edges in $G[S] - \text{BRANCH}$ is bounded (see the definition of TC decomposition). According to Lemma 15, $G[S_{j_m}(t_{m+1})]$ is 2 edge-connected, a contradiction. Hence, $G[R_i]$ is 2 edge-connected.

Next, we show that for any nonempty set $S' \subseteq S^* - R_i$, $G[S' \cup R_i]$ is not 2 edge-connected. A similar discussion to the proof of Case 1 can be used to prove this result.

From the above analysis, $G[R_i]$ is a 2E-block.

Lemma 18. *After executing line 29 in Algorithm 4,* $EB = \{G[R_1], G[R_2], \cdots, G[R_M]\}$ *contains exactly all 2E-blocks in* $G[S]$.

Proof. Lemma 17 shows that each $G[R_i] \in EB$ is a 2E-block. Next, we need to prove only that except for $G[R_1], G[R_2], \cdots, G[R_M]$, there are no other 2$E$-blocks in $G[S]$. In contrast, assume that there is another 2E-block $G[R^*] \notin EB$ with $R^* = \{x_1, x_2, \cdots, x_k\} \subseteq S (k \ge 1)$ in $G[S]$. Consider the following cases. □

Case 1. $|R^*| = 1$. Note that $S = R_1 \cup R_2 \cdots \cup R_M$; then, there exists some $i \in \{1, 2, \cdots, M\}$ such that $R^* = \{x_1\} \subseteq R_i$.

Input: A 2 edge-connected graph $G = (V, E)$, a colored subgraph $G[S]$ of G and $Eblock\_ID(x)$, where $x \in S \subseteq V$.
Output: The set EB consisting of all 2E-blocks in $G[S]$.
1 Construct a BFS tree T in $G[S]$, and let $BRANCH$ be the set of edges in T. Let $E_{done} = \varnothing$, $S_{undo} = S$.
2 **for** $i = 1$ to $|S|$ **do**
3 Select a node u with the smallest ID_u in S_{undo}.
4 **if** $E(S, u) - BRANCH = \varnothing$ **then**
5 **if** u is a white node **then**
6 Color u blue, and $Eblock\_ID(u) = ID_u$.
7 **end if**
8 **else**
9 Let $E_u = \{(u, v) \mid (u, v) \in E(S, u) - (BRANCH \cup E_{done})\}$
10 **for** each edge $e \in E_u$ **do**
11 Find a simple circle C, whose edges are all in the edge set $BRANCH \cup \{e\}$, and let $F(C)$ be the set of all nodes in C.
12 **if** there are no red nodes in $F(C)$ **then**.
13 Let K_1 be the minimum ID of nodes in $F(C)$.
14 For each $w \in F(C)$, let $Eblock\_ID(w) = K_1$.
15 Color all nodes in $F(C)$ red.
16 **else**
17 Let $Eblock\_ID(F(C)) = \{Eblock\_ID(w) \mid w \in F(C)\}$. Let $Q = Eblock\_ID(F(C))$ and $K_2 = \min\{x \mid x \in Q\}$.
18 For each $w \in F(C)$, $Eblock\_ID(w) = K_2$.
19 Color all nodes in $F(C)$ red.
20 **for** each $p \in (Q - \{K_2, \infty\})$ **do**
21 For each node $w \in S$ with $Eblock\_ID(w) = p$, $Eblock\_ID(w) = K_2$.
22 **end for**
23 **end if**
24 **end for**
25 $E_{done} = E_{done} \cup E_u$.
26 **end if**
27 $S_{undo} = S_{undo} - \{u\}$.
28 **end for**
29 Let $\{EID_1, EID_2, \cdots, EID_M\} = \{Eblock\_ID(x) \mid x \in S\}$ and $R_i = \{x \in S \mid Eblock\_ID(x) = EID_i\}(i = 1, 2, \cdots, M)$. Let $EB = \{G[R_1], G[R_2], \cdots, G[R_M]\}$
30 **return** EB.

Algorithm 4: Calculate 2E-blocks.

According to the definition of a 2E-block, $G[R^*]$ is not a 2E -block, a contradiction.

Case 2. $|R^*| \geq 2$. Consider the following subcases.

Case 1. There is $R_i (i \in \{1, 2, \cdots, M\})$ such that $R^* \subseteq R_i$. According to the definition of a 2E-block, $G[R^*]$ is not a 2 E-block, a contradiction.

Case 2. There is not an $R_i (i \in \{1, 2, \cdots, M\}$ such that $R^* \subseteq R_i$. Note that $R^* \subseteq S = R_1 \cup R_2 \cdots \cup R_M$. Then, $\exists x_i, x_j \in R^*$ and $R_p, R_q \in EB$ such that $x_i \in R_p$ and $x_j \in R_q (p \neq q)$. According to the assumption, R^* is a 2E-block, which implies that $G[R^*]$ is 2 edge-connected paths; thus, there exist 2 edge-disjoint paths between x_1 and x_2 that form a simple circle. Hence, $G[R^* \cup R_p \cup R_q]$ is 2 edge-connected, a contradiction to the assumption that $G[R^*]$ is a 2E-block.

Thus, $EB = \{G[R_1], G[R_2], \cdots, G[R_M]\}$ contains exactly all 2E-blocks in $G[S]$.

Lemma 19. *In each iteration of the while-loop (lines 3-11) in Algorithm 3, before the execution of line 4, there must exist a shortest path $L_k([S_0 \cup connector])(k \leq 2)$ in G that connects at least 2 different 2E-blocks in $G[S_0 \cup connector]$.*

Proof. According to line 3, before the execution of line 4, there exist at least two 2E-blocks in $G[\bar{S}]$, where $\bar{S} = S_0 \cup$ connector. In the following discussions, $\bar{S} = S_0 \cup$ connector is a set obtained before the execution of line 4. Let EC_1, $EC_2, \cdots, EC_r (r \geq 2)$ be all the 2E-blocks in $G[\bar{S}]$. □

First, we prove that in G, there must exist an $L_k(\bar{S})$ path that connects at least 2 different 2E-blocks. Since $r \geq 2$, there must exist a bridge in $G[\bar{S}]$. Let e be a bridge in $G[\bar{S}]$; then, $\exists x, y \in \bar{S}$ such that all paths between x and y contain bridge e, which implies that there are no edge-disjoint paths between x and y in $G[\bar{S}]$. On the other hand, since G is a 2 edge-connected graph, there are two edge-disjoint paths between x and y in G (Lemma 12). Let H be the shortest path from x to y in G that does not contain bridge e and has at least one node outside $G[\bar{S}]$. For convenience, for $u, v \in V$, we use $P(u, v)$ to denote a path from u to v and $\bar{P}(u, v)$ to denote $P(u, v) - \{u, v\}$, and we let $H = x \longrightarrow \cdots \longrightarrow x_1 \longrightarrow \bar{P}(x_1, x_2) \longrightarrow x_2 \longrightarrow \cdots \longrightarrow x_3 \longrightarrow \bar{P}(x_3, x_4) \longrightarrow x_4 \cdots \longrightarrow x_t \longrightarrow \bar{P}(x_t, x_{t+1}) \longrightarrow x_{t+1} \longrightarrow \cdots \longrightarrow y$, where each $\bar{P}(x_i, x_{i+1})(1 \leq i \leq t)$ is nonempty and its nodes are outside $G[\bar{S}]$ and the nodes in $H - \cup_{i=1}^{t} \bar{P}(x_i, x_{i+1})$ are in $G[\bar{S}]$. We claim that there exists some $\bar{P}(x_i, x_{i+1})(1 \leq i \leq t)$ such that x_i and

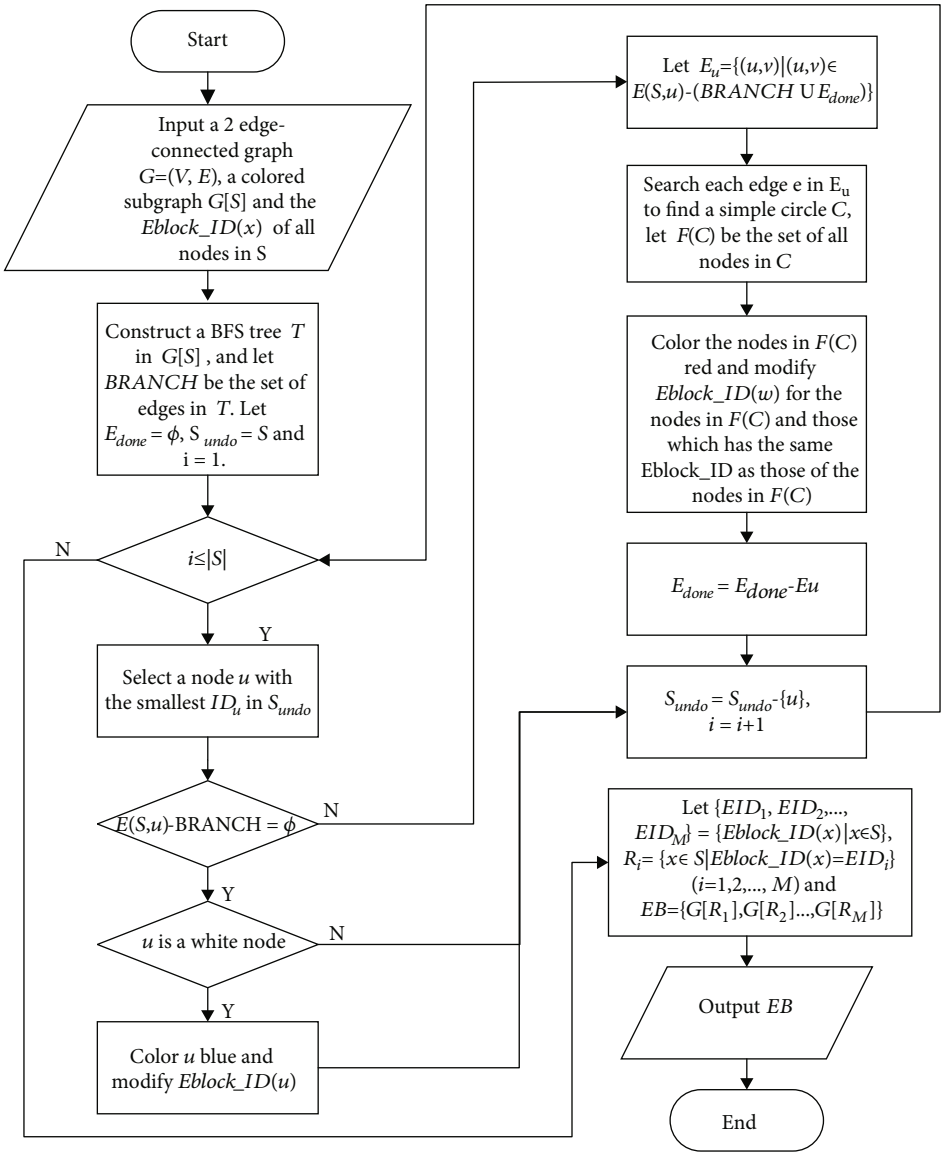


Figure 5: The flow chart of Algorithm 4.

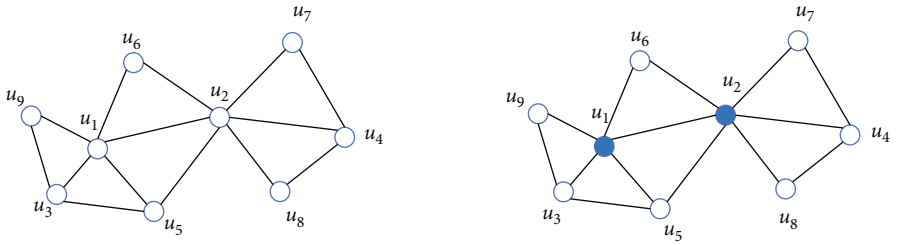
x_{i+1} belong to two different 2E-blocks in $G[\bar{S}]$. In contrast, assume that for each $\bar{P}(x_i, x_{i+1})(1 \le i \le t)$, x_i, x_{i+1} belongs to the same 2E-block, say $EC_l(1 \le l \le t)$ in $G[\bar{S}]$ (see Figure 8). According to the definition of a 2E-block, there is a path $Q(x_i, x_{i+1})$ from x_i to x_{i+1} in the 2E-block EC_l. Now, using $Q(x_i, x_{i+1})$ to replace $x_i \longrightarrow \bar{P}(x_i, x_{i+1}) \longrightarrow x_{i+1}(1 \le i \le t)$ in H, we obtain a new path H' in $G[\bar{S}]$ that does not contain bridge e, a contradiction. Hence, the claim is true.

Now, we suppose that $\bar{P}(x_i, x_{i+1})$ satisfies the condition that x_i and x_{i+1} belong to two different 2E-blocks in $G[\bar{S}]$, say $x_i \in EC_1$, $x_{i+1} \in EC_2$. Let $\bar{P}(x_i, x_{i+1}) = x_{i1} \longrightarrow x_{i2} \longrightarrow \cdots \longrightarrow x_{ik}$; then, $x_i \longrightarrow x_{i1} \longrightarrow x_{i2} \longrightarrow \cdots \longrightarrow x_{ik} \longrightarrow x_{i+1}$ is an $L_k(\bar{S})$. Next, we prove that there is an $L_m(\bar{S})(m \le 2)$ in G by induction on k, the number of nodes in $\bar{P}(x_i, x_{i+1})$. When $k \le 2$, the result is clearly true.

When $k = 3$, since $\bar{S}$ is a dominating set, there is a node in $\bar{S}$, say z, such that $z \in N(x_{i2})$. If $z \notin EC_1$, then $x_i \longrightarrow x_{i1} \longrightarrow x_{i2} \longrightarrow z$ is an $L_2(\bar{S})$; the result is true. If $z \in EC_1$, then $z \longrightarrow x_{i2} \longrightarrow x_{i3} \longrightarrow x_{i+1}$ is an $L_2(\bar{S})$; the result is true.

Assume that when $k = n$, the result is true. Now, we consider the situation $k = n + 1$. Then, $\bar{P}(x_i, x_{i+1}) = x_{i1} \longrightarrow x_{i2} \longrightarrow \cdots \longrightarrow x_{i(n+1)}$. Similarly, since $\bar{S}$ is a dominating set, there is a node in $\bar{S}$, say z, such that $z \in N(x_{i2})$. If $z \notin EC_1$, then $x_i \longrightarrow x_{i1} \longrightarrow x_{i2} \longrightarrow z$ is an $L_2(\bar{S})$; the result is true. If $z \in EC_1$, then $z \longrightarrow x_{i2} \longrightarrow \cdots \longrightarrow x_{i(n+1)} \longrightarrow x_{i+1}$ is an $L_n(\bar{S})$. According to the inductive assumption, we have that the result is true.

Lemma 20. *In Algorithm 3, let connector$_j$ be the set connector obtained after executing the jth iteration of the*



(a) Illustration of a network graph $G = (V, E)$ with 9 nodes (b) The CDS S_{CDS} obtained by calling ACC11 for G

(c) The connected 2 edge-dominating set S_0 obtained by calling Algorithm 2 for G (d) The $(2, 2)$-ECDS S_1' obtained by calling Algorithm 3 for G

Figure 6: Illustration of the steps of LLZ20.

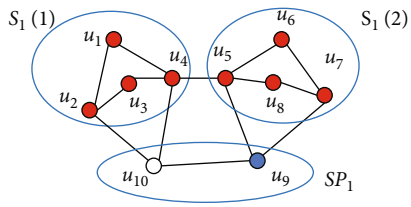


Figure 7: An example illustrating TC decomposition.

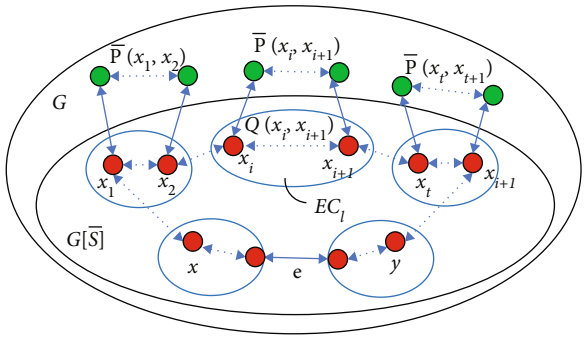


Figure 8: An example illustrating path $\bar{P}(x_i, x_{i+1})$ and path $Q(x_i, x_{i+1})$.

while-loop (lines 3-11) $(1 \le j \le |S_0|)$ *and let* EB_j *be the set consisting of all 2E-blocks in* $G[S_0 \cup connector_j]$ *(*EB_0 *denotes the set consisting of all 2E-blocks in* $G[S_0]$*). Then,* $|EB_j| < |EB_{j-1}|$.

Proof. Let L_k^j be the $L_k(S_0 \cup \text{connector}_j)(k = 1 \text{ or } 2)$ in the jth iteration of the *while*-loop (lines 3-11) in Algorithm 3 and $EB_j = \{M_1^j, M_2^j, \cdots, M_s^j\}$. Without loss of generality, let M_1^{j-1}, M_2^{j-1} be these two 2E-blocks in EB_{j-1}, which are connected by L_k^j, and let x_1 and x_2 be the two endpoints of L_k^j with $x_i \in V(M_i^{j-1})(i = 1, 2)$. Since $G[S_0 \cup \text{connector}_{j-1}]$ is connected, there is a path P_1 between x_1 and x_2 in $G[S_0 \cup \text{connector}_{j-1}]$. Note that $E(L_k^j) \cap E(P_1) = \varnothing$. Therefore, there is a simple circle containing x_1 and x_2 in $L_k^j \cup P_1$, denoted by C_1. After calling Algorithm 4 to recompute the 2E-blocks of $G[S_0 \cup \text{connector}_j]$, the 2$E$-block_IDs of nodes in $M_1^j \cup M_2^j \cup V(L_k^j)$ are the same.

Let $EID_{C_1} = \{Eblock\_ID(x) \mid x \in V(C_1)\}$. Clearly, $|EID_{C_1}| \ge 2$ before executing the jth iteration of the *while*-loop (lines 3-11). Let $A_j = \{x \in S_0 \cup \text{connector}_j \mid Eblock\_ID(x) \in EID_{C_1}\}$.

In the jth iteration of the *while*-loop (lines 3-11) in Algorithm 3, after executing line 6 or line 9, which calls Algorithm 4 to recompute the 2E-blocks in $G[S_0 \cup \text{connector}_j]$, each node $x \in A_j$ is set to have the same Eblock_ID value, while the Eblock_ID values of the nodes in $(S_0 \cup \text{connector}_j) - A_j$ are unchanged. Thus, the number of different Eblock_IDs of the nodes in $G[S_0 \cup \text{connector}_j]$ is less than that of different Eblock_IDs of the nodes in $G[S_0 \cup \text{connector}_{j-1}]$. According to line 29 in Algorithm 4 and Lemma 18, we conclude that the number of different 2E-blocks in $G[S_0 \cup \text{connector}_j]$ is less than that in $G[S_0 \cup \text{connector}_{j-1}]$; in other words, $|EB_j| < |EB_{j-1}|$.

Next, we use the above preliminaries to prove that S_1' generated by Algorithm 3 is a $(2, 2)$-ECDS of G. In fact, according to Lemmas 18 and 19, we know that in Algorithm 3, the *while*-loop (lines 3-11) can be executed only if the number of 2E-blocks in $G[S_0 \cup \text{connector}]$ is more than 1. Thus, according to Lemma 20, we conclude that after executing Algorithm 3, there is only one 2E-block in $G[S_0 \cup$

connector]. Furthermore, according to the definition of a 2E -block, $G[S_0 \cup \text{connector}]$ is 2 edge-connected blocks. This implies that $S_1 = S_0 \cup \text{connector}$ (line 12) is a $(2,2)$-ECDS of G. Note that $VD \subseteq D$ and $D \subseteq S_0$ (see *while*-loop (lines 8-14) and line 15 in Algorithm 2); thus, $VD \subseteq S_0$. According to the *for*-loop (lines 13-17), it is easy to see that S_1' (line 18) is 2 edge-connected.

Let $S^* \subseteq VD$ be the set of nodes deleted in the *for*-loop (lines 13-17); then, after Algorithm 3 is executed, $S_1' = S_1 - S^*$ and $S^* \subseteq VD \subseteq S_1$, which imply that $V - S_1' = S^* \cup (V - S_1)$. Next, we prove that S_1' is a 2 edge-dominating set. Equivalently, we need to prove only that for $u \in V - S_1'$, u is 2 edge-dominated by S_1'. Note that $S^* \cap (V - S_1) = \varnothing$; thus, $u \in S^*$ or $u \in V - S_1$. Consider the following cases. □

Case 1. $u \in S^*$. Since $S^* \subseteq VD$ is an independent set and $G[S_1] = G[S_1' \cup S^*]$ is 2 edge-connected, u is 2 edge-dominated by S_1', where S_1' is the S_1' obtained after the *for*-loop (lines 13-17) is executed.

Case 2. $u \in V - S_1$. Note that $V - S_1 = V - (S_1 \cup S^*) \subseteq V - (S_0 \cup S^*)$ and that, according to Corollary 14, u is 2 edge-dominated by $S_0 - S^*$. On the other hand, since $S_0 - S^* \subseteq S_1 - S^* = S_1'$, u is 2 edge-dominated by S_1'.

Overall, S_1' is a 2 edge-dominating set. Thus, S_1' (line 18) is a $(2,2)$-ECDS of G.

According to the above discussions, we obtain the following theorem.

Theorem 21. *The output of Algorithm 1 is a $(2,2)$-ECDS of G.*

4.2. Analysis of the Performance Ratio and Time Complexity. In the above subsection, we provided proof that a $(2,2)$-ECDS must be obtained by our algorithms. In this subsection, we prove that LLZ20 is an approximation algorithm and then present its performance ratio.

In line 1 of Algorithm 1, for a given graph G, we utilize ACC11 in [32] to construct a CDS in G. In this process, a CDS is calculated in 2 steps. In the first step, an MIS is obtained; in the second step, some nodes outside the MIS are added to the MIS such that it becomes a CDS S_{CDS}. Das et al. [32] proved that the size of S_{CDS} is no more than $6.78 * \text{opt}_G + 8.12$, where opt_G is the MCDS size in G.

Lemma 22. *After executing line 2 in Algorithm 1, $|S_0| \leq 10.17 * opt_G + 12.68$, where opt_G is the MCDS size in G.*

Proof. According to lines 7-14 of Algorithm 2, D is an MIS in $G[B_0]$. Since $D \subseteq S_{\text{MIS}}$, $|D| \leq |S_{\text{MIS}}|$, where S_{MIS} is an MIS in G. According to [30], $|S_{\text{MIS}}| \leq 3.39 * \text{opt}_G + 4.56$. Hence, $|D| \leq 3.39 * \text{opt}_G + 4.56$.

According to line 15 of Algorithm 2, we obtain the following inequalities:

$$\begin{aligned} |S_0| \leq |S_{\text{CDS}}| + |D| &\leq 6.78 * \text{opt}_G + 8.12 + 3.39 * \text{opt}_G + 4.56 \\ &= 10.17 * \text{opt}_G + 12.68. \end{aligned} \tag{1}$$

□

Lemma 23. *After executing Algorithm 3, $|\text{connector}| \leq 2 * (|S_0| - 1)$.*

Proof. Let m be the number of 2E-blocks in line 2 of Algorithm 3. Clearly, $m \leq |S_0|$. According to Lemma 20, the number of iterations of the *while*-loop (lines 3-11) in Algorithm 3 is no more than $m - 1$. Note that in each iteration of the *while*-loop (lines 3-11), one or two new nodes are added to connector. Hence, $|\text{connector}| \leq 2 * (m - 1) \leq 2 * (|S_0| - 1)$. □

Theorem 24. *After executing line 3 in Algorithm 1, $|S_1'| \leq 30.51 * opt + 36.04$, where opt is the minimum $(2,2)$-ECDS size in G.*

Proof. Let opt_G be the MCDS size in G; then, $\text{opt}_G \leq \text{opt}$. According to Algorithm 3 and Lemmas 22 and 23, we have the following inequalities:

$$\begin{aligned} |S_1'| \leq |S_1| = |S_0| + |\text{connector}| &\leq 3|S_0| - 2 \leq 30.51 * \text{opt}_G + 36.04 \\ &\leq 30.51 * \text{opt} + 36.04. \end{aligned} \tag{2}$$

□

In the following paragraphs, we discuss the time complexity of LLZ20. The overall time cost of LLZ20 includes three components: (1) the time cost of computing a CDS S_{CDS} by ACC11, (2) the time cost of computing a 2 edge-dominating CDS S_0 by Algorithm 2, and (3) the time cost of constructing a $(2,2)$-ECDS S_1' by Algorithm 3.

Denote the number of nodes in a given input G by n. In the first phase, according to [32], the time cost of constructing a CDS S_{CDS} by ACC11 is $O(n^3)$.

In the second phase, the time cost of calculating the set B_0 of the nodes that have only one dominator in S_{CDS} is $O(n)$ (the *for*-loop (lines 2-6)). Moreover, the time cost of finding an MIS in $G[B_0]$ is $O(n^2)$ (the *while*-loop (lines 8-14)). Hence, the time cost of constructing a 2 edge-dominating CDS S_0 by Algorithm 2 is $O(n + n^2) = O(n^2)$.

In the third phase, since Algorithm 4 is called to compute the 2E-blocks in the subgraph induced by the updated $S_0 \cup \text{connector}$, the time complexity of Algorithm 4 must be analyzed. In Algorithm 4, the time cost of constructing a BFS tree T is $O(n^2)$. In the iterations of the *for*-loop (lines 2-28), the time cost of computing all 2E-blocks in $G[S]$ based on T is $O(n^3)$. Hence, the time complexity of Algorithm 4 is $O(n^2 + n^3) = O(n^3)$. Now, we discuss the time cost of using

Table 1: Parameters and their values used in the simulations.

Parameters	Definitions	Values
n	The size of network	[10, 250]
r	Transmission range of each node	25 m or 35 m
k	Data packet size	256 bytes
E_{elec}	Energy dissipated in electronics for 1-bit transmission and reception	50 nJ/bit
E_{tr}	Energy consumed in power amplifier in two-ray propagation model	0.0013 pJ/bit/m^4
β	Ratio of energies consumed in reception and idle listening modes	0.8
E_0	Initial energy of each node	1 J
m	Number of packets per round	10

Algorithm 3 to obtain the $(2,2)$-ECDS S_1'. In Algorithm 3, iteration of the *while*-loop (lines 3-11) is equal to $O(n)$, and the overall behavior of line 2, line 6, and line 9 is similar. Each computes the $2E$-blocks in the subgraph induced by the updated $S_0 \cup$ connector by calling Algorithm 4, which implies that all of their time costs are $O(n^3)$. Consequently, the time cost of adding $L_k(S_0 \cup \text{connector})$ paths to connect a 2 edge-dominating set S_0 to form a $(2,2)$-ECDS (*while*-loop (lines 3-11)) is $O(n^4)$. In addition, the time cost of deleting some redundant nodes from S_1 (lines 13-17) is $O(n^4)$. Hence, in the third phase, the time complexity of constructing a $(2,2)$-ECDS S_1' by Algorithm 3 is $O(n^3 + n^4 + n^4) = O(n^4)$.

Overall, the time complexity of Algorithm 1 is $O(n^3 + n^2 + n^4) = O(n^4)$.

Theorem 25. *Suppose that the number of nodes in a given input G is n; then, the time complexity of LLZ20 (Algorithm 1) is* $O(n^4)$.

5. Simulation and Analysis

In the previous sections, we proposed an approximation algorithm LLZ20 to compute a $(2,2)$-ECDS, a new CDS with edge fault tolerance, in a given network graph. At the same time, we conducted a theoretical analysis of the performance ratio of LLZ20. In this section, we perform simulation experiments to evaluate the performance of our algorithm in terms of the size of the CDS, running time, and success rate and compare our algorithms with the existing related algorithms proposed by [25, 26]. In the rest of this section, we refer to the algorithm in [25] as SYWH08 and that in [26] as SZZW16. To objectively compare the performances of these three algorithms (LLZ20, SYWH08, and SZZW16), we utilize the algorithm proposed in [32], which, to the best of our knowledge, is the algorithm with the minimum performance ratio in terms of the size of the CDS, to generate a CDS. Notably, the authors of [26] proposed an algorithm to calculate a $(2,m)$-CDS based on a known $(1,m)$-CDS; however, they did not present a specific method to generate such a $(1,m)$-CDS. In our simulation experiments, we use the algorithms proposed by [48] to calculate a $(1,m)$-CDS based on a CDS obtained by [32]'s algorithm. The key parameters used in the simulations are listed in Table 1. In the simulation experiments, the employed hardware and software are an Intel Core i5 CPU 2.56 GHz, 16 GB DRAM, 64-bit Windows 10, and the Programming IDE is MATLAB.

5.1. Effects of the Number of Nodes. In the experimental setup, for convenience, we choose a virtual space with a 100×100 experimental area. We let the transmission radius r of each node be 25 m and set the network size n to 10,20,30,40, $\cdots$, 250.

First, we provide a brief introduction to the generation process of candidate networks. Consider n nodes randomly distributed in the above virtual space. After a WSN has been determined, the program judges whether it is 2 connected: if yes, the network is chosen as a candidate network for the experiments; otherwise, it is abandoned. The procedure is repeated 100 times, and the candidate networks are saved. In the simulation experiment in this subsection, the chosen network graphs are 2 connected but not 2 edge-connected graphs. The main reasons for this selection are as follows. On the one hand, when the chosen network graph is a 2 connected graph, all three algorithms can generate a $(2,2)$-ECDS; however, when the chosen network graph is a 2 edge-connected graph, only our algorithm can guarantee the generation of a $(2,2)$-ECDS (SYWH08 and SZZW16 cannot). On the other hand, to objectively compare the performance of our algorithm with that of SYWH08 and SZZW16 in terms of CDS size and algorithm running time, the three algorithms should be able to generate a $(2,2)$-ECDS in the same experimental environment.

For each candidate network, G, LLZ20 (SYWH08 and SZZW16, respectively) is run to calculate a $(2,2)$-ECDS, and the running time and resulting CDS size are recorded. These operations are repeated for all candidates, and the averages are calculated as the simulation result. Figures 9 and 10 exhibit the simulation results for LLZ20, SYWH08, and SZZW16 in terms of CDS size and algorithm running time, respectively.

Figure 9 presents the relationship between the simulation curves for the $(2,2)$-ECDS size of these three algorithms. The resulting $(2,2)$-ECDS size increases as the network size increases for all three algorithms. This result is reasonable because the larger the network size is, the greater the number of nodes needed to dominate the network. Moreover, our algorithm (LLZ20) outperforms SYWH08 in terms of the CDS size, and the CDS size

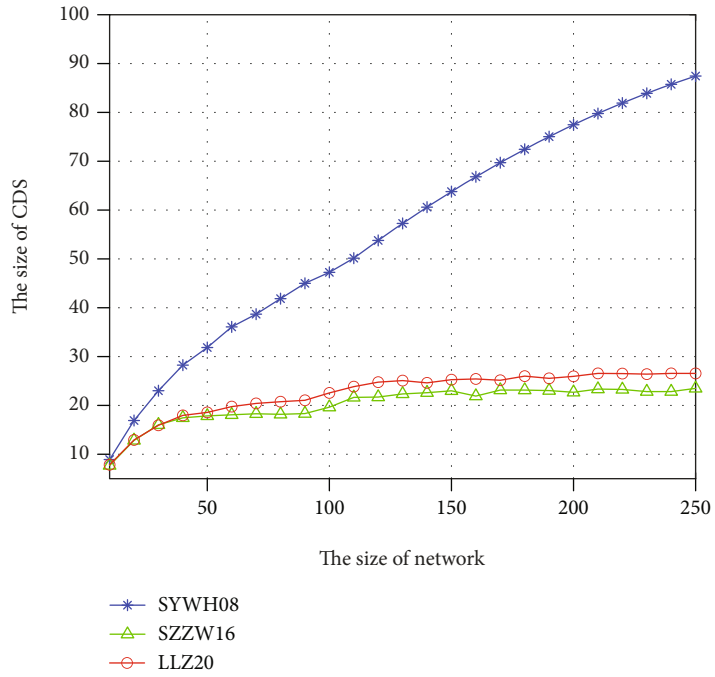


Figure 9: Comparison of LLZ20, SYWH08, and SZZW16 in terms of the CDS size.

difference between LLZ20 and SYWH08 increases as the network size increases. This is because SYWH08 is designed for a $(2, 2)$-CDS, which is a $(2, 2)$-ECDS, while LLZ20 is specifically designed for a $(2, 2)$-ECDS. This difference may cause the number of nodes added to the $(1, 2)$-ECDS to form the resulting $(2, 2)$-ECDS in SYWH08 to be greater than that in LLZ20. According to Figure 10, the average running time of our algorithm is almost the same as that of SYWH08. Strictly speaking, the average running time of SYWH08 is slightly less than that of our algorithm. This slightly longer running time is tolerable because the performance of our algorithm is clearly better than that of SYWH08 in terms of CDS size, and the difference in average running time is within 5 s, even when the network size n is 250.

Figure 9 shows that the size of the $(2, 2)$-ECDS obtained by our algorithm is slightly larger than that obtained by SZZW16 for different network sizes. This result is acceptable for the following reasons. On the one hand, compared with SZZW16, our algorithm requires a shorter time to obtain the simulation results (see the discussion in the next paragraph). On the other hand, as mentioned in Introduction, some network graphs have an edge fault-tolerant CDS and do not contain any node fault-tolerant CDS. To guarantee that the simulation result of SZZW16 is obtained, 2-connected network graphs have to be chosen as the candidate networks for simulations in the simulation experiment. More specifically, compared with SZZW16, to guarantee that an expected CDS is obtained, our algorithm is designed to be suitable for a larger number of wireless networks; see the discussions regarding the next simulation for details. In addition, the difference between the size of the CDS obtained by our algorithm and that of the CDS obtained by SZZW16 is small (approximately 2).

Next, we consider the simulation results shown in Figure 10, which show that the performance of our algorithm is better than that of SZZW16 in terms of average running time.

Figure 10 shows that the average running time of our algorithm is less than that of SZZW16 and that the average running time of SZZW16 increases superlinearly with increasing network size, while the average running time of our algorithm increases slightly with increasing network size. When $n \geq 150$, the average running time of SZZW16 exceeds 570 s and is more than 57 times that of our algorithm. This result is reasonable. In fact, in the process of generating the $(2, 2)$-ECDS, to select some nodes to add to the $(1, 2)$-ECDS to form a $(2, 2)$-ECDS, SZZW16 needs to compute the potential function value of each node in the network to determine whether it can be selected, and the computation of the potential function value for a node is very complex. Therefore, this process has a high time cost. Consequently, it is natural that this time cost increases as the network size increases. However, in the process of selecting some nodes to add to a $(1, 2)$-ECDS to form a $(2, 2)$-ECDS, for a node outside the $(1, 2)$-ECDS, our algorithm only needs to make a simple comparison of the node neighbor information between this node and other nodes outside the $(1, 2)$-ECDS to determine whether it can be selected, which entails only a small time cost.

On the basis of the above analysis, we can conclude that among SYWH08, SZZW16, and LLZ20, our algorithm is the most suitable algorithm to compute a $(2, 2)$-ECDS in network graphs based on the comprehensive consideration of the CDS size and algorithm running time.

5.2. Effects of the Network Density. In this subsection, we present simulations carried out to investigate the effects of

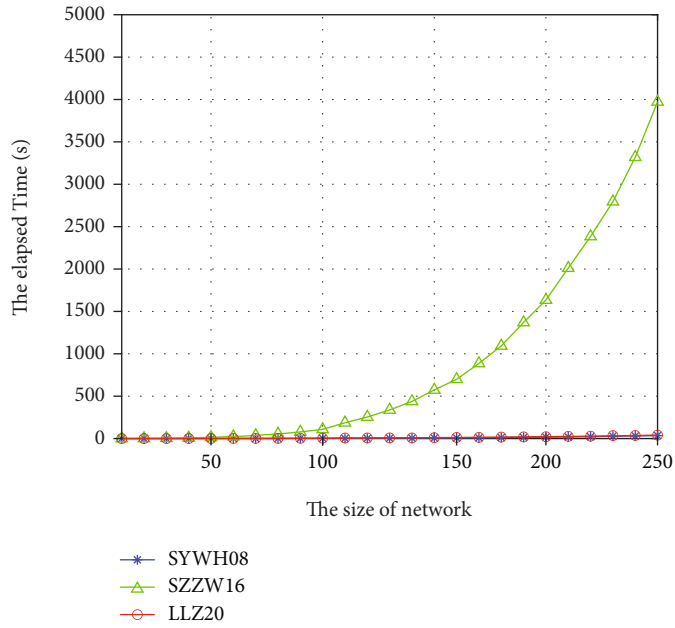


Figure 10: Comparison of LLZ20, SYWH08, and SZZW16 in terms of running time.

the network density to compare the performance of the three algorithms. Specifically, we investigate how a change in network density affects the CDS size for these three algorithms. To obtain simulation results showing how the CDS size changes as the network density changes, we fix the network size to $n = 250$ and the node transmission radius to 25 m and vary the area from 40 m × 40 m to 140 m × 140 m. In this experiment, we randomly generate 100 candidate networks, which are 2-connected, in an area with the size varying as described. For each candidate network, we run each of the three algorithms to obtain a $(2, 2)$-ECDS and record the CDS size. For a given density, we repeat this operation for 100 candidate networks and take the average of the simulation results.

Figure 11 compares the performance of the three algorithms in terms of the CDS size. From Figure 11, it is easily found that the CDS size increases as the network area size increases; in other words, the CDS becomes larger as the network density decreases. This is reasonable because the sparser the network is, the fewer nodes are dominated by a node in the network. In additional, from Figure 11, we can see that LLZ20 outperforms SYWH08 for each area size (i.e., network density) and shows similar performance to SZZW16. Specifically, the CDS generated by LLZ20 is smaller than that generated by SYWH08 by at least 57, and the difference increases as the network density increases (when the network area size is 40 m × 40 m, the difference is approximately 81). On the other hand, the size of the CDS obtained by LLZ20 is generally close to that obtained by SZZW16, and as the density increases, their difference decreases. In particular, when the network area size is 40 m × 40 m, this difference is approximately 2. This is acceptable for a reason similar to that mentioned when discussing the above simulation results, namely, the time complexity of SZZW16 is far greater than that of our algorithm, LLZ20. In other words, LLZ20 outperforms SZZW16 on average.

5.3. Comparison of the Success Rate. As mentioned in the beginning of Subsection 4.1, in theory, for any given 2 edge-connected network graphs, our algorithm can always output a $(2, 2)$-ECDS, while SYWH08 and SZZW16 may fail. If an algorithm cannot output an expected $(2, 2)$-ECDS for the problem of finding the minimum $(2, 2)$-ECDS for a given 2 edge-connected network, the algorithm is useless for this problem. In other words, in a randomly generated 2 edge-connected network, for the problem of finding the minimum $(2, 2)$-ECDS, the probability that the considered algorithm successfully outputs a $(2, 2)$-ECDS is an important factor in evaluating its performance. We call this probability the *success rate* of the considered algorithm. In this subsection, we compare the performance of three algorithms (LLZ20, SYWH08, and SZZW16) in terms of the success rate by means of a simulation experiment. For convenience, T_s/T_a is utilized to denote the considered algorithm's success rate, where T_a is the number of total experimental networks and T_s is the number of experimental networks in which a $(2, 2)$-ECDS is successfully output by the algorithm.

In the experimental setup, we also select 100 candidates, where the selection process is similar to that in the above subsection, except for the condition used to determine whether a randomly generated network graph is a candidate. In this simulation, this condition is the following: "After such a WSN has been determined, the program judges whether it is 2 edge-connected; if yes, this network is chosen as a candidate network for experiments; otherwise, it is abandoned."

For each candidate network G, we run LLZ20 to calculate a $(2, 2)$-ECDS and record whether a $(2, 2)$-ECDS can be output. After the results are obtained for 100 candidate networks, we calculate the success rate T_s/T_a, where $T_a = 100$. We perform

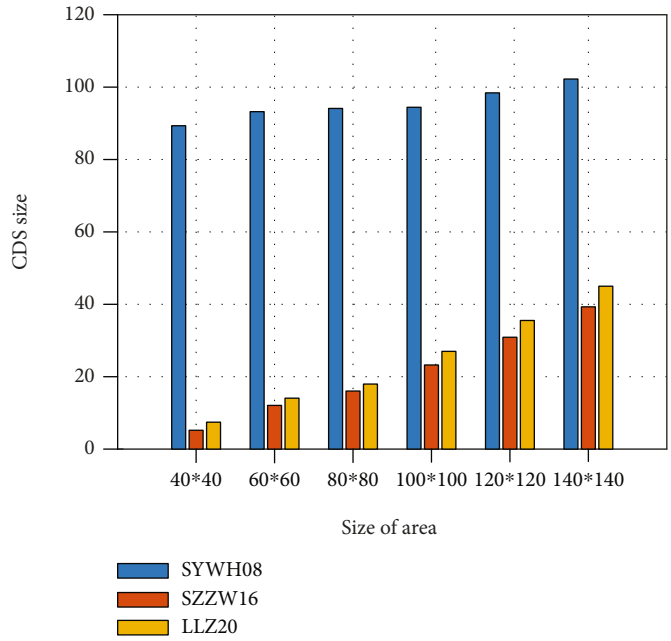


Figure 11: Comparison of LLZ20, SYWH08, and SZZW16 in different network density.

Table 2: The success rates in the case of $r = 25$.

Network size	10	20	30	40	50	60	70	80	90	100
SYWH08	74%	90%	94%	96%	97%	97%	98%	98%	99%	99%
SZZW16	74%	90%	94%	96%	97%	97%	98%	98%	99%	99%
LLZ20	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

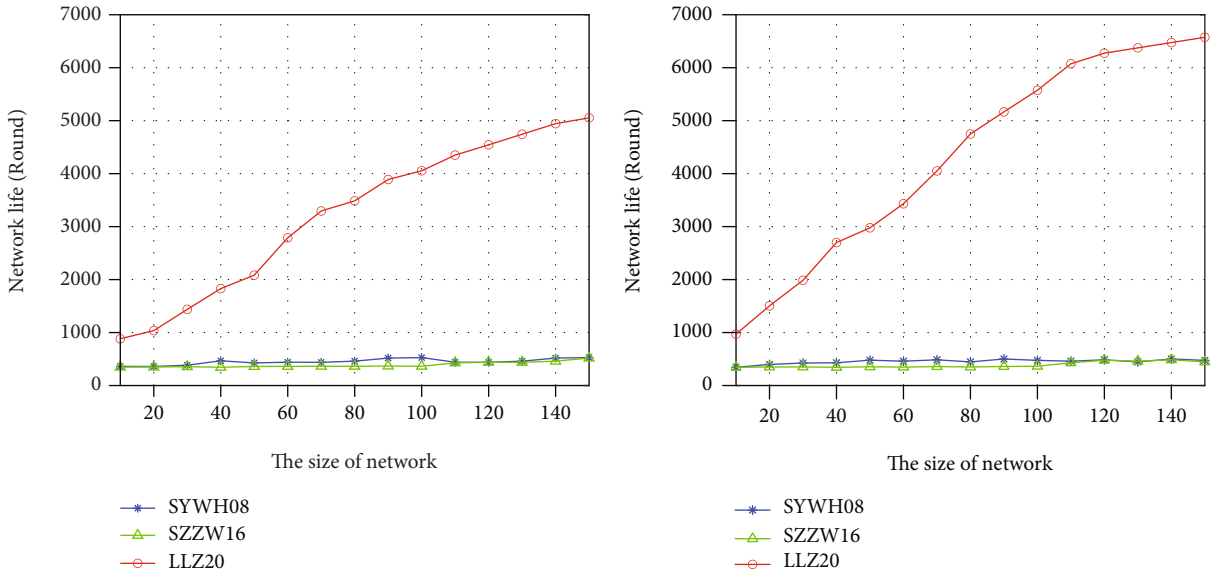
similar operations for SYWH08 and SZZW16 to obtain the corresponding success rates. The simulation results for these three algorithms are shown in Table 2.

According to Table 2, the success rate of our algorithm is 100% for all network sizes. This result is reasonable because in the theoretical analysis (Section 4), for any 2 edge-connected network graphs, we strictly prove that our algorithm must output a $(2, 2)$-ECDS. Table 2 also shows that the success rates of SYWH08 and SZZW16 are less than 100%, which is reasonable for the following reasons. On the one hand, when SYWH08 (SZZW16) runs in a network that is not 2 connected, it fails to generate a $(2, 2)$-2ECDS. On the other hand, the network graphs in the simulation experiments are 2 edge-connected but not necessarily 2 connected graphs (according to the definitions of a 2 edge-connected graph and 2 connected graph, a 2 connected graph must be a 2 edge-connected graph, but the inverse is not true). Moreover, Table 2 indicates that when the network size n decreases, the number of candidate networks in which SYWH08 and SZZW16 fail to output an expected $(2, 2)$-ECDS becomes larger. It is natural because when the network size n decreases or the number of nodes deployed randomly in a given area decreases, the density of the network becomes sparser, which increases the probability that cut nodes exist in the network. Consequently, the probability that this network graph is not a 2 connected graph increases.

5.4. Comparison of the Network Lifetime. In this subsection, we compare SYWH08, SZZW16, and LLZ20 in terms of the network lifetime. Note that for each of these algorithms, the first step requires the employment of other algorithms to generate a $(1, 1)$-CDS. Therefore, to objectively compare the network lifetime performance of these three algorithms, we utilize the $(1, 1)$-CDS generated by the algorithm proposed in [49], which is an approximation algorithm with favorable performance in terms of the network lifetime, as the CDS generated in the first step. For the experimental setup, we choose a virtual space with a 100×100 experimental area. We let the transmission radius r of each node to be 25 m (or 35 m) and set the network size n to 10,20,30,40, $\cdots$, 150. Similar to Subsection 4.1, 100 candidate networks are generated for this simulation experiment.

For convenience, we use the model proposed in [50] as the energy consumption model for our simulation. The energy consumption model mainly includes the following features: (1) the energy consumption for sending a packet is $E_{TX} = E_{elec} * k + E_{tr} * k * r^4$, (2) the energy consumption for receiving a packet is $E_{RX} = E_{elec} * k$, and (3) the energy consumption in the idle listening state is $E_{RX} = \beta * E_{elec} * k$. The definitions of the notations E_{elec}, E_{tr}, k, and β are given in Table 1. The energy consumption for constructing or reconstructing a CDS for these three algorithms is ignored.

To measure the performance of an algorithm K in terms of the network lifetime, we need to introduce an index,



(a) Transmission range $r = 25$

(b) Transmission range $r = 35$

Figure 12: Comparison of LLZ20, SYWH08, and SZZW16 in terms of network lifetime.

called the number of rounds and denoted by Round_K, which is defined as follows. Based on the above energy consumption model, for the algorithm of interest, the following actions are considered to constitute one round: a sender s and receiver r are randomly selected, and s successfully sends m packets to r via the shortest path in the CDS generated by the algorithm under the condition that whole network contains no nodes with an energy of 0. To compute Round_K, some additional assumptions are needed, as follows: (1) the initial energy of every node in the network is 1 J; (2) all nodes in the CDS are in the idle listening state; (3) the nodes outside the CDS are in the sleeping state, so their energy consumption is ignored; (4) each packet is 256 bytes; (5) all nodes on this shortest path in the CDS (except s and r) both receive and transmit these m packets; (6) for each node on this shortest path in the CDS, its neighbors outside the shortest path only receive these m packets and do not forward them to their neighbors; and (7) a new CDS needs to be generated by the algorithm K once the residual energy of any node in the CDS is reduced to 50%.

We take the average number of rounds for 100 networks as the simulation result for each algorithm, and the simulation results are shown in Figure 12.

Figure 12 shows that the performance of our algorithm LLZ20 is much better than that of SZZW16 or SYWH08 in terms of the network lifetime. Specifically, the number of rounds (network lifetime) under SZZW16 or SYWH08 is 360 fewer than that under LLZ20 for all network sizes. This result is reasonable because when a new CDS is constructed, LLZ20 uses the greedy strategy based on the remaining node energy to select the nodes to constitute its CDS, while both SZZW16 and SYWH08 use other strategies to select the nodes for their CDSs; this implies that compared with LLZ20, SZZW16 (or SYWH08) has a higher probability of selecting some nodes selected in previous rounds to perform message routing in the current round, which, in turn, implies that the number of rounds under LLZ20 may be greater than that under SZZW16 (SYWH08).

Moreover, it can be observed that for LLZ20, the network lifetime increases as the network size increases. This is also reasonable. In the process of reconstructing a CDS via a greedy strategy based on energy, as the network size increases, the number of nodes with high residual energy available to form a CDS also increases. This means that compared with those in a smaller network, the new CDS nodes in a larger network will have higher residual energy after the same number of rounds. In contrast, from Figure 12, we can see that for SZZW16 or SYWH08, the network lifetime changes little with increasing network size. This is easily understood. Because some of the nodes chosen to form a new CDS during CDS reconstruction are selected in ways other than an energy-based greedy strategy in these two algorithms, this implies that there is a higher probability that nodes selected in previous rounds will be selected for message routing in the current round. In other words, in these two algorithms, it is expected that some nodes will be repeatedly selected to form different CDSs regardless of the network size, and the residual energy of these nodes will heavily impacts the lifetime of the whole network.

6. Conclusion

In this paper, to address the problem of the edge fault tolerance of VBs in WSNs, we introduced the concept of the $(2, 2)$-ECDS, a kind of CDS with edge fault tolerance, and proposed an effective algorithm for computing a $(2, 2)$-ECDS in a 2 edge-connected network graph. The proposed algorithm is proven to be an approximation algorithm with a performance ratio of 30.51. By means of simulations, we compared the performance of our algorithm with that of existing related algorithms in terms of CDS size, algorithm running time, and algorithm success rate; the simulation results show

that our algorithm is more suitable than the existing related algorithms for constructing a CDS with edge tolerance. In recent years, how to improve the energy efficiency of networks has been investigated in several studies [39, 51]. In our future work, we will investigate the construction of VBs by considering edge fault tolerance and energy efficiency in heterogeneous WSNs and develop new algorithms for constructing (k, m)-ECDSs for any k and m with $m \geq k \geq 2$ in homogeneous WSNs.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grant no. 61862003 and in part by the Natural Science Foundation of Guangxi Zhuang Autonomous Region of China under grant no. 2018GXNSFDA281052.

References

[1] H. Yetgin, K. T. K. Cheung, M. el-Hajjar, and L. Hanzo, "A survey of network lifetime maximization techniques in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 828–854, 2017.

[2] O. Kaiwartya, A. H. Abdullah, Y. Cao et al., "Virtualization in wireless sensor networks: fault tolerant embedding for internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 571–580, 2018.

[3] W. Liang, C. Ma, M. Zheng, and L. Luo, "Relay node placement in wireless sensor networks: from theory to practice," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1602–1613, 2021.

[4] W. Zhang, J. Liang, and X. Liang, "On the computation of virtual backbones with fault tolerance in heterogeneous wireless sensor networks," *IEEE Transactions on Mobile Computing*, p. 1, 2021.

[5] K. Li, X. Gao, F. Wu, and G. Chen, "A constant factor approximation for d -hop connected dominating set in three-dimensional wireless networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, pp. 4357–4367, 2019.

[6] H. Mostafaei, A. Montieri, V. Persico, and A. Pescapé, "An efficient partial coverage algorithm for wireless sensor networks," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, pp. 501–506, Messina, Italy, 2016.

[7] H. Mostafaei, M. Chowdhury, R. Islam, and H. Gholizadeh, "Connected P-percent coverage in wireless sensor networks based on degree constraint dominating set approach," in *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 157–160, Cancun, Mexico, 2015.

[8] W. Wang, B. Liu, D. Kim, D. Li, J. Wang, and W. Gao, "A new constant factor approximation to construct highly fault-tolerant connected dominating set in unit disk graph," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 18–28, 2017.

[9] J. Wang, Y. Gao, C. Zhou, R. Simon Sherratt, and L. Wang, "Optimal coverage multi-path scheduling scheme with multiple mobile sinks for WSNs," *Computers, Materials & Continua*, vol. 62, no. 2, pp. 695–711, 2020.

[10] J. Wang, C. Ju, Y. Gao, A. K. Sangaiah, and G. J. Kim, "A PSO based energy efficient coverage control algorithm for wireless sensor networks," *Computers, Materials & Continua*, vol. 56, no. 3, pp. 433–446, 2018.

[11] J. Wang, Y. Gao, X. Yin, F. Li, and H. J. Kim, "An enhanced PEGASIS algorithm with mobile sink support for wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 9472075, 9 pages, 2018.

[12] K. Vijayalakshmi, "Global levy flight of cuckoo search with particle swarm optimization for effective cluster head selection in wireless sensor network," *Intelligent Automation & Soft Computing*, vol. 26, no. 2, pp. 303–311, 2019.

[13] A. Hady, "Duty cycling centralized hierarchical routing protocol with content analysis duty cycling mechanism for wireless sensor networks," *Computer Systems Science and Engineering*, vol. 35, no. 5, pp. 347–355, 2020.

[14] S. Tabatabaei, "A novel fault tolerance energy-aware clustering method via social spider optimization (SSO) and fuzzy logic and mobile sink in wireless sensor networks (WSNs)," *Computer Systems Science & Engineering*, vol. 35, no. 6, pp. 477–494, 2020.

[15] Y. Shi, Z. Zhang, Y. Mo, and D. Z. du, "Approximation algorithm for minimum weight fault-tolerant virtual backbone in unit disk graphs," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 925–933, 2017.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, USA, 1978.

[17] F. Dai and J. Wu, "On constructing k-connected k-dominating set in wireless networks," in *19th IEEE International Parallel and Distributed Processing Symposium*, Denver, CO, USA, April 2005.

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

[19] W. Wang, B. Liu, D. Kim, D. Li, J. Wang, and Y. Jiang, "A better constant approximation for minimum 3-connected m-dominating set problem in unit disk graph using Tutte decomposition," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1796–1804, Hong Kong, China, April 2015.

[20] T. Fukunaga, "Approximation algorithms for highly connected multi-dominating sets in unit disk graphs," *Algorithmica*, vol. 80, no. 11, pp. 3270–3292, 2018.

[21] B. Liu, W. Wang, D. Kim, Y. Li, S. S. Kwon, and Y. Jiang, "On practical construction of quality fault-tolerant virtual backbone in homogeneous wireless networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 412–421, 2018.

[22] Z. Zhang, J. Zhou, Y. Mo, and D.-Z. Du, "Performance-guaranteed approximation algorithm for fault-tolerant connected dominatingset in wireless networks," in *IEEE INFOCOM*

2016 - The 35th Annual IEEE International Conference on Computer Communications, pp. 1–8, San Francisco, CA, USA, April 2016.

[23] 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 *2010 Proceedings IEEE INFOCOM*, pp. 1–9, San Diego, CA, USA, March 2010.

[24] T. Fukunaga, "Constant-approximation algorithms for highly connected multi-dominating sets in unit disk graphs," *Algorithmica*, vol. 80, no. 1, pp. 3270–3292, 2017.

[25] W. Shang, F. Yao, P. Wan, and X. Hu, "On minimum m -connected k-dominating set problem in unit disc graphs," *Journal of Combinatorial Optimization*, vol. 16, no. 2, pp. 99–106, 2008.

[26] Y. Shi, Y. Zhang, Z. Zhang, and W. Wu, "A greedy algorithm for the minimum 2 –connected m-fold dominating set problem," *Journal of Combinatorial Optimization*, vol. 31, no. 1, pp. 136–151, 2016.

[27] F. Wang, M. T. Thai, and D.-Z. Du, "On the construction of 2-connected virtual backbone in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1230–1237, 2009.

[28] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.

[29] D.-Z. Du, R. L. Graham, P. M. Pardalos, P.-J. Wan, W. Wu, and W. Zhao, "Analysis of greedy approximations with nonsubmodular potential functions," in *Proceedings 19th ACM-SIAM symposium on discrete algorithms*, San Francisco, California, USA, January 2008.

[30] Y. L. Du and H. W. Du, "A new bound on maximum independent set and minimum connected dominating set in unit disk graphs," *Journal of Combinatorial Optimization*, vol. 30, no. 4, pp. 1173–1179, 2015.

[31] Y. Li, M. Thai, F. Wang, C. W. Yi, P. J. Wan, and D. Z. du, "On greedy construction of connected dominating sets in wireless networks," *Wireless Communications and Mobile Computing*, vol. 5, no. 8, p. 932, 2005.

[32] A. Das, C. Mandal, and C. Reade, "An improved greedy construction of minimum connected dominating sets in wireless networks," in *2011 IEEE Wireless Communications and Networking Conference*, Cancun, Mexico, May 2011.

[33] P.-J. Wan, K. M. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 9no. 2, pp. 141–149, 2002.

[34] P.-J. Wan, L. Wang, and F. Yao, "Two-phased approximation algorithms for minimum CDS in wireless ad hoc networks," in *2008 The 28th International Conference on Distributed Computing Systems*, Beijing, China, July 2008.

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

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

[37] W. Wu, H. Du, X. Jia, Y. Li, and S. C. H. Huang, "Minimum connected dominating sets and maximal independent sets in unit disk graphs," *Theoretical Computer Science*, vol. 352, no. 1-3, pp. 1–7, 2006.

[38] M. Min, H. du, X. Jia, C. X. Huang, S. C. H. Huang, and W. Wu, "Improving construction for connected dominating set with Steiner tree in wireless sensor networks," *Journal of Global Optimization*, vol. 35, no. 1, pp. 111–119, 2006.

[39] J. Mohanty, C. Mandal, and C. Reade, "Distributed construction of minimum connected dominating set in wireless sensor network using two-hop information," *Computer Networks*, vol. 123, no. 4, pp. 137–152, 2017.

[40] Y. Wu, F. Wang, M. T. Thai, and Y. Li, "Constructing k-connected m-dominating sets in wireless sensor networks," in *MILCOM 2007 - IEEE Military Communications Conference*, pp. 1–7, Orlando, FL, USA, 2007.

[41] B. Liu, W. Wang, D. Kim et al., "On approximating minimum 3-connected m-dominating set problem in unit disk graph," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2690–2701, 2016.

[42] A. E. Zonouz, L. Xing, V. M. Vokkarane, and Y. Sun, "A time-dependent link failure model for wireless sensor networks," in *2014 Reliability and Maintainability Symposium*, Colorado Springs, CO, 2014.

[43] X. Fu, H. Yao, and Y. Yang, "Modeling cascading failures for wireless sensor networks with node and link capacity," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7828–7840, 2019.

[44] U. Draz, T. Ali, S. Yasin, U. Waqas, and U. Rafiq, "Towards formalism of link failure detection algorithm for wireless sensor and actor networks," in *2019 International Conference on Engineering and Emerging Technologies (ICEET)*, pp. 1–6, Lahore, Pakistan, 2019.

[45] H. Xu, X. Bai, P. Liu, and Y. Shi, "Hierarchical fusion estimation for WSNs with link failures based on Kalman-consensus filtering and covariance intersection," in *2020 39th Chinese control conference (CCC)*, pp. 5150–5154, Shenyang, China, 2020.

[46] X. Liang, J. Liang, and W. Zhang, "Constructingd-robust connected dominating sets in wireless sensor networks with unstable transmission ranges," *IEEE Transactions on Communications*, vol. 69, no. 1, pp. 398–415, 2021.

[47] K. Rosen, *Discrete mathematics and its application*, WCB/McGraw-Hill, 2007.

[48] J. Zhou, Z. Zhang, W. Wu, and X. Xing, "A greedy algorithm for the fault-tolerant connected dominating set in a general graph," *Journal of Combinatorial Optimization*, vol. 28, no. 1, pp. 310–319, 2014.

[49] D. Kim, Y. Wu, Y. Li, F. Zou, and D. Du, "Constructing minimum connected dominating sets with bounded diameters in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 2, pp. 147–157, 2009.

[50] V. Raghunathan, C. Schurgers, Sung Park, and M. B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 40–50, 2002.

[51] O. Dagdeviren, K. Erciyes, and S. Tse, "Semi-asynchronous and distributed weighted connected dominating set algorithms for wireless sensor networks," *Computer Standards & Interfaces*, vol. 42, pp. 143–156, 2015.