IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 27, 2020, accepted September 13, 2020, date of publication September 18, 2020, date of current version October 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3024964

Network Simplification and K-Terminal Reliability Evaluation of Sensor-Cloud Systems

YUCHANG MO[1], (Senior Member, IEEE), MIN LIANG[1], LIUDONG XING[2], (Senior Member, IEEE), JINPING LIAO[1], AND XULI LIU[1]

[1]Fujian Province University Key Laboratory of Computational Science, School of Mathematical Sciences, Huaqiao University, Quanzhou 362021, China
[2]Department of Electrical and Computer Engineering, University of Massachusetts at Dartmouth, Dartmouth, MA 02747, USA

Corresponding author: Yuchang Mo (yuchangmo@sina.com)

This work was supported in part by the NSF of China under Grant 61972156, in part by the Program for Innovative Research Team in Science and Technology in Fujian Province University, and in part by the Quanzhou High-Level Talents Support Plan under Grant 2017ZT012.

ABSTRACT The sensor-cloud system (SCS) integrates sensors, sensor networks, and cloud for managing sensors, collecting and processing data, and decision-making based on data processed. Though the SCS has received tremendous attention from both academia and industry because of its numerous exciting applications, it still faces the challenge in reliability. The reliability of an SCS is generally referred to as the ability to perform required functions for a given period of time. This work is focused on the K-terminal reliability of an SCS, which is concerned with the successful communication between all pairs of network nodes belonging to a pre-specified subset K. The increased complexity and scale of real-life SCSs require new efficient techniques to evaluate their K-terminal reliability. In this work we make novel contributions by proposing a network simplification method that can effectively remove all redundant network edges and vertices, leading to a significantly reduced network model for accurate and efficient K-terminal reliability analysis. The method is based on graph decomposition and reconstruction through articulation vertices. Empirical studies show that the proposed simplification method integrated with the binary-decision-diagrams based evaluation algorithm can significantly speed up K-terminal reliability analysis of large real-life SCSs.

INDEX TERMS Sensor-cloud system, K-terminal reliability, network simplification, binary decision diagram.

I. INTRODUCTION

The sensor-cloud system (SCS) integrates sensors, sensor networks, and cloud for managing sensors, collecting and processing data, and subsequent decision-making [1], [2]. The integration provides flexible, low-cost, and reconfigurable platforms for monitoring and controlling objects in various applications, including emerging applications of the Internet of Things (IoT), machine to machine, and cyber-physical systems.

Network models are widely applied for analyzing many SCS systems. Because networks in SCSs usually display a high degree of tolerance to random failures and attacks due to redundant paths, the SCS network reliability analysis is a challenging task. There are two classes of reliability metrics for SCS systems: network-level metrics and application-level metrics.

Network-level metrics concern the reliability or resilience of the whole network. For example, two metrics of isolation time and probability of isolation were studied in [3], where the isolation time is the expected delay before a vertex is forcefully isolated from the network (i.e., the time before all neighbors of the vertex are simultaneously in the failed state) and the probability of isolation is the probability of the isolation occurring within the vertex's lifetime.

Application-level metrics often concern the successful communication among a specified set of vertices through at least one fault-free path or tree [4]. For example, the K-terminal reliability of a network system is concerned with the successful communication between all pairs of vertices of a subset K. The K-terminal network reliability evaluation is known to be an NP-hard problem [5]. In this article, we focus on the problem of K-terminal network reliability evaluation.

The associate editor coordinating the review of this manuscript and approving it for publication was Datong Liu.

One common class of traditional exact evaluation of K-terminal reliability is based on the enumeration of all minimal paths (MPs) or minimal cuts (MCs). Once all MPs/MCs are identified, the network reliability can be evaluated using different techniques, like inclusion-exclusion (IE) or sum of disjoint products (SDPs) [6]. The major limitation of the MPs/MCs-based methods is as the number of network edges becomes large, the number of MPs/MCs can become too large to be evaluated using the IE or SDPs expression. Recently, the recursive decomposition algorithms have been proposed by integrating a shortest path search algorithm and a disjoint path identification process to improve the efficiency of network reliability calculation [7]. In [8], a matrix-based system reliability method was proposed based on efficient matrix manipulations and MCs identification. Later, the methods in [5]–[8] were integrated and applied to estimate the reliability of a real-world network in [9], where a smaller number of non-minimal disjoint sets were identified to fully represent the network connectivity. Although the methods in [5]–[9] improved the computational efficiency, they still need to deal with the exponentially increasing number of MPs/MCs. In [10], in order to improve the computational efficiency of MP/MC identification or inclusion-exclusion calculation, a universal generating function (UGF)-based recursive procedure was proposed to evaluate the network reliability. The UGF-based method was improved by Yeh through some reduction techniques in [11]. However, the UGF-based methods still suffer from computational inefficiency on large-scale networks. A survey of recent developments of UGF-based methods in the field of network reliability estimation can be found in [12].

To solve large-scale networks, the Monte Carlo Simulation (MCS) approaches can be used because their computational efficiency depends largely on the convergence of probability not the size of networks. However, it has been shown that in highly reliable networks the crude MCS method is impractical, because the probability of network failure is a rare-event probability [13], [14]. The search for efficient MCS algorithms for highly reliable networks has resulted in a number of variance reduction methods, such as conditional MCS approaches, approximate zero-variance importance sampling [15], and combinations of these [16]. For a survey of these methods see [17]. It should be noted that the statistical errors in the simulation-based approaches may result in slow convergence for achieving acceptable accuracy in low probability estimations and in parameter sensitivity calculations, and this challenge may hamper the rapid reliability estimation for large-scale networks.

In the last few decades, binary decision diagrams (BDD), an extraordinarily efficient method to represent and manipulate Boolean functions [18], [19], [36], have been exploited to model different classes of systems, such as multi-state systems [20], [21], dynamic systems [22], [37], phased-mission systems [23], [24], [38], IoT systems and networks [39], [40], wireless sensor networks [41], [42], and online social networks [43]. Among these efforts, BDD-based approaches for reliability analysis of large-scale networks were proposed in [25]–[29]. Despite the efficiency improvement over the traditional MP/MC based approaches, the BDD-based method can become inefficient or even impractical for analyzing very large networks. Therefore, some researchers have attempted to develop network reduction techniques to simplify the target network for BDD-based network reliability analysis. However, these efforts are still partial; the existing network simplification techniques cannot effectively remove all the redundant vertices. For example, in [25]–[27], a vertex (other than the source and sink) is defined as a redundant vertex if only one another vertex is connected to it. These redundant vertices can be easily identified using the definition and be removed to avoid manipulation of some redundant graph searching. In practice, vertices with more than one neighboring vertex can also be redundant vertices and thus should be removed for efficient network reliability analysis.

In this article,, we focus on this line of research and make novel contributions by proposing a comprehensive network simplification method. The main contributions of our study can be summarized as follows:

1. Based on the concept of articulation vertices, we construct a valid subnetwork from the original network by removing all redundant networking elements (edges and vertices) that have no contribution to the K-terminal reliability to be evaluated. The method is based on graph decomposition and reconstruction through articulation vertices and has polynomial time complexity. Based on the generated valid subnetwork having fewer elements (edges and vertices), the BDD model can be constructed more quickly by avoiding inefficient manipulations involved in the analysis based on the original network.
2. The size of BDD (i.e., the number of non-sink nodes) encoding a network heavily depends on the chosen ordering of the network components. A comprehensive comparative study has been conducted on the high-performance edge ordering for SCS networks. The performance comparisons among existing ordering heuristics are conducted and some improvements on the heuristic performance for SCS networks are proposed.

The rest of this article, is organized as follows: Section II introduces background. Section III is devoted to the development of a simplification method based on articulation vertices for constructing the valid subnetwork and shows the correctness of the proposed simplification method. Experimental data of applying the proposed method to two large real-life SCS network models are presented in Section IV. Finally, Section V gives conclusions.

II. BACKGROUND

In the network reliability analysis, a ***K*-terminal network** is typically modeled using an undirected probabilistic graph $G = (V, E, K)$, with V as the vertex set, $E \subseteq V \times V$ as the edge set and $K \subseteq V$ as a vertex subset. Fig.1 gives

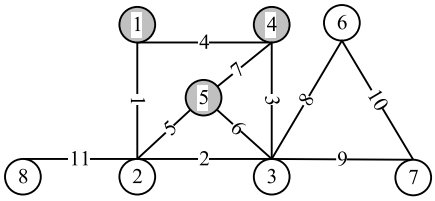


FIGURE 1. An example network (*K* set is denoted by shaded circles).

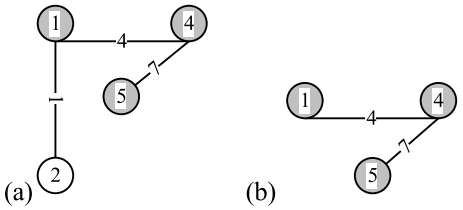


FIGURE 2. Illustrative *K*-trees.

an example network, where $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $E = \{(1, 2), (1, 4), (2, 3), (2, 5), (2, 8), (3, 4), (3, 5), (3, 6), (3, 7), (4, 5), (6, 7)\}$, and $K = \{1, 4, 5\}$.

A. THE CONCEPT OF K-TREE

A K spanning tree, or simply, ***K*-tree** of $G = (V, E, K)$, denoted by kt, is a connected tree in G if all vertices of the K set are connected in kt. A **minimal *K*-tree** of $G = (V, E, K)$, denoted by mkt, is a K-tree containing no subgraphs that are also K-trees. For example, the network in Fig.2(a) is a K-tree of G in Fig.1, but it is not a minimal K-tree because one of its subgraphs as shown in Fig.2(b) is also a K-tree. Fig.3 presents all different minimal K-trees for the example network in Fig.1.

B. K-TERMINAL RELIABILITY

Each edge k of G is subject to failure with a known probability q_k $(0 \leq q_k \leq 1)$ or time-to-failure distribution that can derive q_k. Thus, $p_k = 1 - q_k$ represents the reliability of edge k. We assume that all the edge failure events are statistically independent and all the nodes are perfectly reliable. Let mkt_i, $1 \leq i \leq N$, be a minimal K-tree of a network G with N minimal K-trees. The K-terminal reliability of G, denoted by R_G, can thus be calculated as shown in Equation (1).

$$R_G = \Pr\{\bigvee_{i=1}^{N} mkt_i\} \tag{1}$$

Because mkt_i $(1 \leq i \leq N)$ are not disjoint events, the evaluation of (1) cannot be performed by simply adding the probability of each mkt_i. Instead, several techniques such as the IE, SDPs, and BDDs can be applied. It has been shown that BDDs are generally more efficient than the IE or SDPs based evaluation. Hence BDDs are selected to evaluate the network reliability R_G in this work.

C. VALID SUBNETWRK

According to Equation (1), it can be derived that the K-terminal reliability of G can be correctly calculated from a subgraph of G where edges and vertices that are not included by any minimal K-tree are deleted. Such a subgraph without edges and vertices that make no contribution to the K-terminal reliability being evaluated is referred to as a valid subnetwork in this work. More formally, a connected subgraph in $G = (V, E, K)$, VG, is a **valid subnetwork** *iff* each edge in VG is included by at least one of the minimal K-trees of G. For the same K set and same network G, the set of all minimal K-trees is unique so there is a unique valid subnetwork. However, different K sets of the same network G have different sets of minimal K-trees and thus cause different valid subnetworks. For example, when $K = \{1, 4, 5\}$ the valid subnetwork of G in Fig.1 is shown in Fig.4(a), and when $K = \{3, 8\}$ a different valid subnetwork is shown in Fig.4(b).

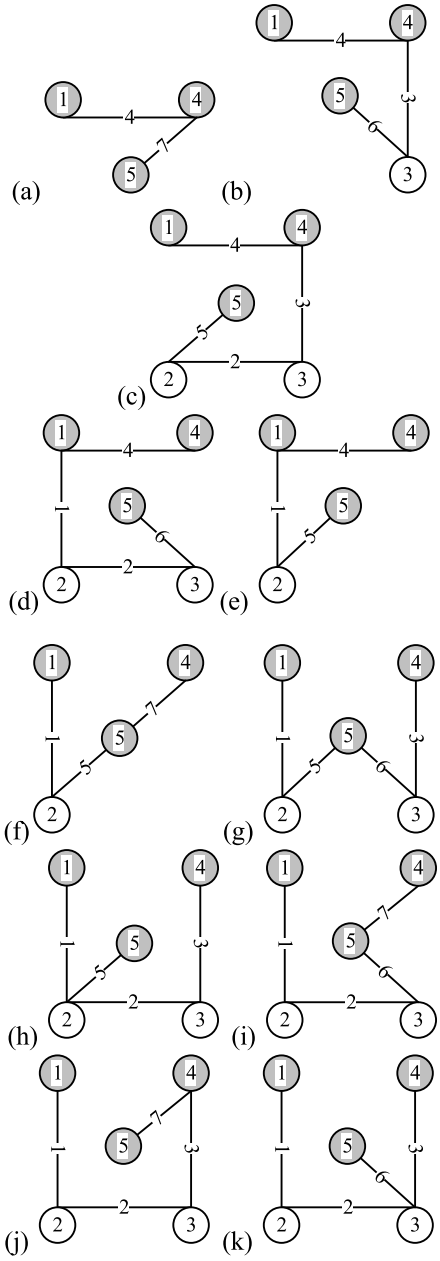


FIGURE 3. Eleven minimal *K*-trees of G in Fig.1.

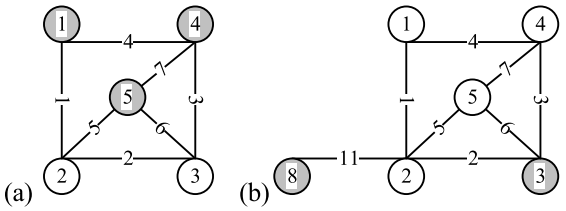


FIGURE 4. Two different valid subnetworks of G in Fig.1. (a) $K = \{1, 4, 5\}$, (b) $K = \{3, 8\}$.

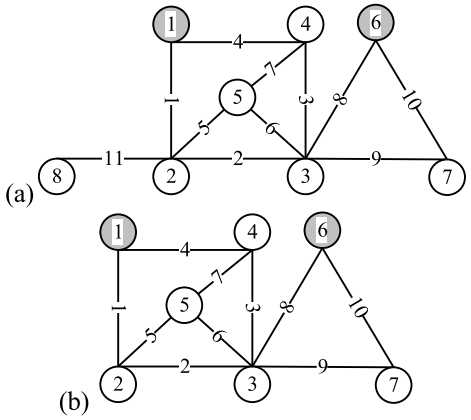


FIGURE 5. Network simplification using the method in [25]–[27]. (a) original network $K = \{1, 6\}$, (b) valid subnetwork $K = \{1, 6\}$.

In general, the valid subnetwork has fewer elements (edges and vertices) than the original network, leading to faster reliability evaluations. Specifically, based on the valid subnetwork, minpaths and mincuts can be enumerated faster or the BDD model can be constructed more quickly by avoiding inefficient manipulation of numerous redundant graph searching and graph expansions. The question that has to be addressed is: **Given a specific$K$-terminal network $G = (V, E, K)$, how to construct its valid subnetwork?**

In [25]–[27], a simple method of removing the redundant vertices (vertices that have only one another vertex connected to) is used. These redundant vertices can be simply identified using the vertex degree (the total number of connections of a vertex). As an illustration, consider the network G in Fig.5(a). When $K = \{1, 6\}$, vertex 8 is a redundant vertex because only vertex 2 is connected to it. In this case, the method in [25]–[27] can simplify the network G into a valid subnetwork as shown in Fig.5(b). However, when $K = \{1, 4, 5\}$ or $K = \{3, 8\}$, the network G cannot be successfully simplified into a valid subnetwork as shown in Fig.4 where the removed redundant vertices are not only vertices with degree 1 (i.e., having one edge).

In this article, we propose a generalized method to construct the valid subnetwork of a K-terminal network by identifying and removing all its possible redundant elements.

III. NETWORK SIMPLIFICATION METHODOLOGY

This section presents the proposed network simplification method based on identification of articulation vertices. The correctness of the proposed method is also verified.

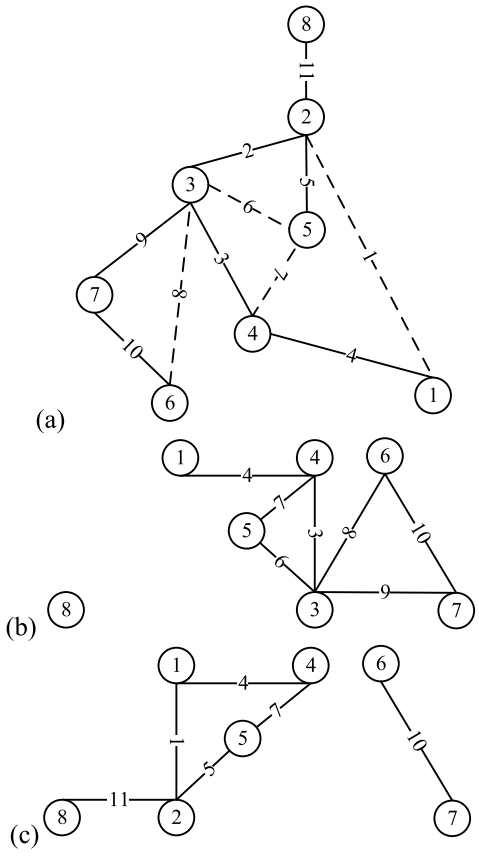


FIGURE 6. Illustration of articulation verteics, (a) DFS tree (b) deletion of articulation vertex 2, (c) deletion of articulation vertex 3.

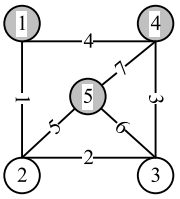


FIGURE 7. The valid subnetwork of G in Fig.1.

A. PROPOSE METHOD

The proposed network simplification method consists of the following three steps.

1) CONSTRUCT AN ARTICULATION VERTICES SET *AV*

An articulation vertex (AV) in a connected network G is a vertex by removing which and associated edges the network is divided into two or more connected subgraphs. Finding articulation vertices can be done by performing a Depth First Search (DFS) [30]. Specifically, in the DFS tree of an undirected network, the root vertex is an articulation vertex *iff* it has more than one child; a leaf vertex is never an articulation vertex; a non-leaf, non-root vertex u is an articulation point *iff* no non-tree edge goes above u from a sub-tree below u.

Consider the network in Fig.1. One possible DFS of the network leads to a tree in Fig 6(a). In this DFS tree, any sub-tree below vertex 2 has no non-tree edge (dotted edges) going above it, and one sub-tree (consists of vertices 7 and 6) below vertex 3 has no non-tree edge (dotted edges) going above it.

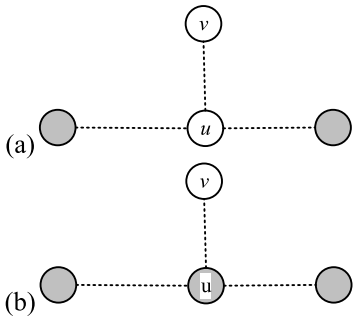


FIGURE 8. mkt has a branch ended by v (a) articulation vertex u not in *K* set; (b) articulation vertex u in *K* set.

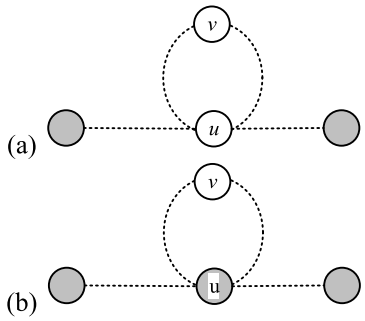


FIGURE 9. Mkt has a loop includes v (a) articulation vertex u in *K* set; (b) articulation vertex u not in *K* set.

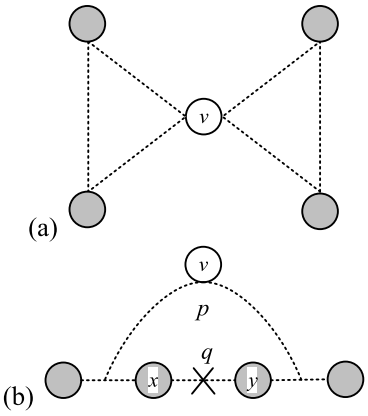


FIGURE 10. Any vertex v in the valid subnetwork is not redundant; (a) v is an articulation vertex; (b) v is not an articulation vertex.

Thus, this network has two articulation vertices, i.e., vertices 2 and 3. If vertex 2 is deleted, the network is broken into two connected subgraphs as shown in Fig.6(b); if vertex 3 is deleted, the network is broken into two connected subgraphs as shown in Fig.6(c).

2) CONSTRUCT A REDUNDANT VERTICES SET *RV*

For each articulation vertex v in the AV, we first delete v and associated edges from G to generate two or more connected subgraphs. For each connected subgraph cc_i that does not inlcude any vertex from the K set, we collect all the vertices in cc_i denoted by $V(cc_i)$, and add them to RV, i.e., $RV = RV \cup V(cc_i)$.

Consider the example network in Fig.1 with $K = \{1, 4, 5\}$. After the processing of articulation vertex 2, we get $RV = \{8\}$, and after the processing of articulation vertex 3, we get $RV = \{6, 7, 8\}$.

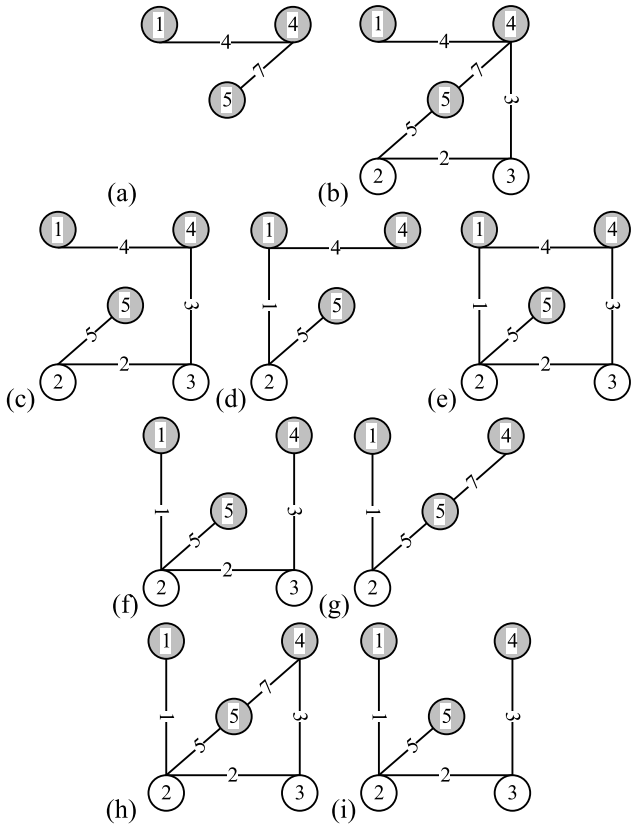


FIGURE 11. Three examples for the second case using the network in Fig.1 with $K = \{1, 4, 5\}$; (a) mkt1, (b) v = 3, x = 4, y = 5, p = {4-3-2-5}, q = {4-5} (c) mkt1′; (d) mkt2, (e) v = 3, x = 1, y = 4, p = {1-2-3-4}, q = {1-4}, (f) mkt2′; (g) mkt3, (h) v = 3, x = 4, y = 5, p = {4-3-2-5}, q = {4-5}, (i) mkt3′.

3) CONSTRUCT THE VALID SUBNETWORK

The valid subnetwork can be simply achieved by deleting all redundant vertices in RV and their associated edges from G. Given that the K set in original network G is connected, the obtained valid subnetwork will be a connected graph according to the above three steps.

Consider the example network in Fig.1 with $K = \{1, 4, 5\}$. By deleting redundant vertices in $RV = \{6, 7, 8\}$, we get a valid subnetwork as shown in Fig.7. We can easily check that it includes all edges in the eleven minimal K-trees in Fig.3.

B. CORRECTNESS OF PROPOSE METHOD

In this section, we show that the subnetwork generated by the above three-step method is a valid subnetwork defined in Section II.C.

Firstly, we prove by contradiction that for any vertex in RV it is not included in any minimal K-tree, i.e., it is redundant for the K-terminal communications. Suppose that there is a minimal K-tree *mkt* that includes a vertex v in RV. According to step 2 in Section III.A, vertex v is in the connected subgraph cc related to articulation vertex u which has no vertex in the K set. There are only two possible cases:

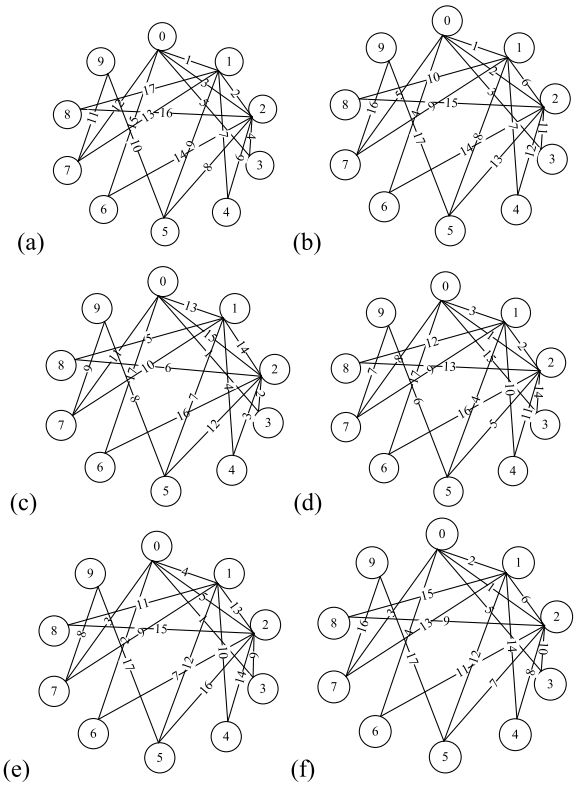


FIGURE 12. Ordering results (a) DFS, (b) BFS, (c) IDDFS, (d) DDDFS, (e) IDBFS, (f) DDBFS.

1) the minimal K-tree mkt has a branch ended by v as shown in Fig.8(a) or Fig.8(b). This case is impossible because v is not in the K set and can be delete from mkt;
2) mkt has a loop which connects v and its corresponding articulation vertex u, as shown in Fig.9(a) or Fig.9(b). This case is also impossible because the minimal K-tree cannot have a loop.

Therefore, all the vertices in RV are truly redundant vertices.

Secondly, we prove that for any vertex v appearing in the final valid subnetwork, it is included by at least one of the minimal K-trees, i.e., it is not redundant. There are two possible cases:

1) v is an articulation vertex in the valid subnetwork, as shown in Fig. 10(a), and each connected subgraph cc related to articulation vertex v includes at least one vertex in K (otherwise, cc will be deleted according to step 2 in Section III.A), thus any minimal K-tree will pass v;
2) v is not an articulation vertex. For this second case, let mkt be a minimal K-tree not including v. Given that the valid subnetwork is a connected graph and v is survived in step 3, we can easily find a path p (including v) by adding which to mkt it will have one loop, and in this loop there will be a path q (not including v) that connects two vertices x and y in the K set (q is required to have no other vertices from the K set other than x and y). We can then add path p and delete path q from x to y in mkt, as shown in Fig,10(b) to get a new minimal K-tree mkt' that includes v. To further illustrate various scenarios in this second case, we give three examples using the network in Fig.1 as shown in Fig.11.

IV. APPLICATION

In this section, we apply the proposed simplification method to two SCS benchmarks. To illustrate the effectiveness of the proposed method, the BDD-based evaluation method is then applied to both the simplified network and the original network for K-terminal reliability analysis.

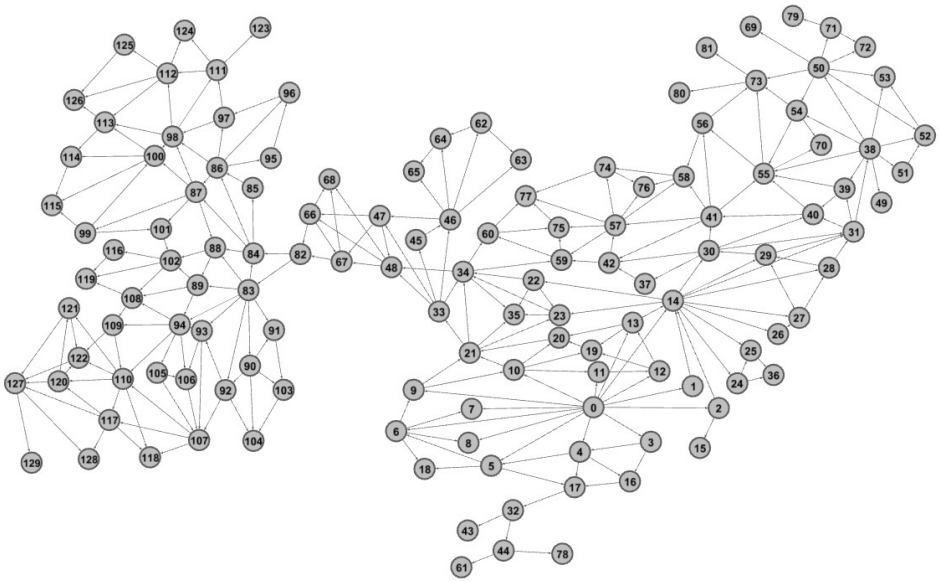


FIGURE 13. The first benchmark SCS deployed in vertical industry services.

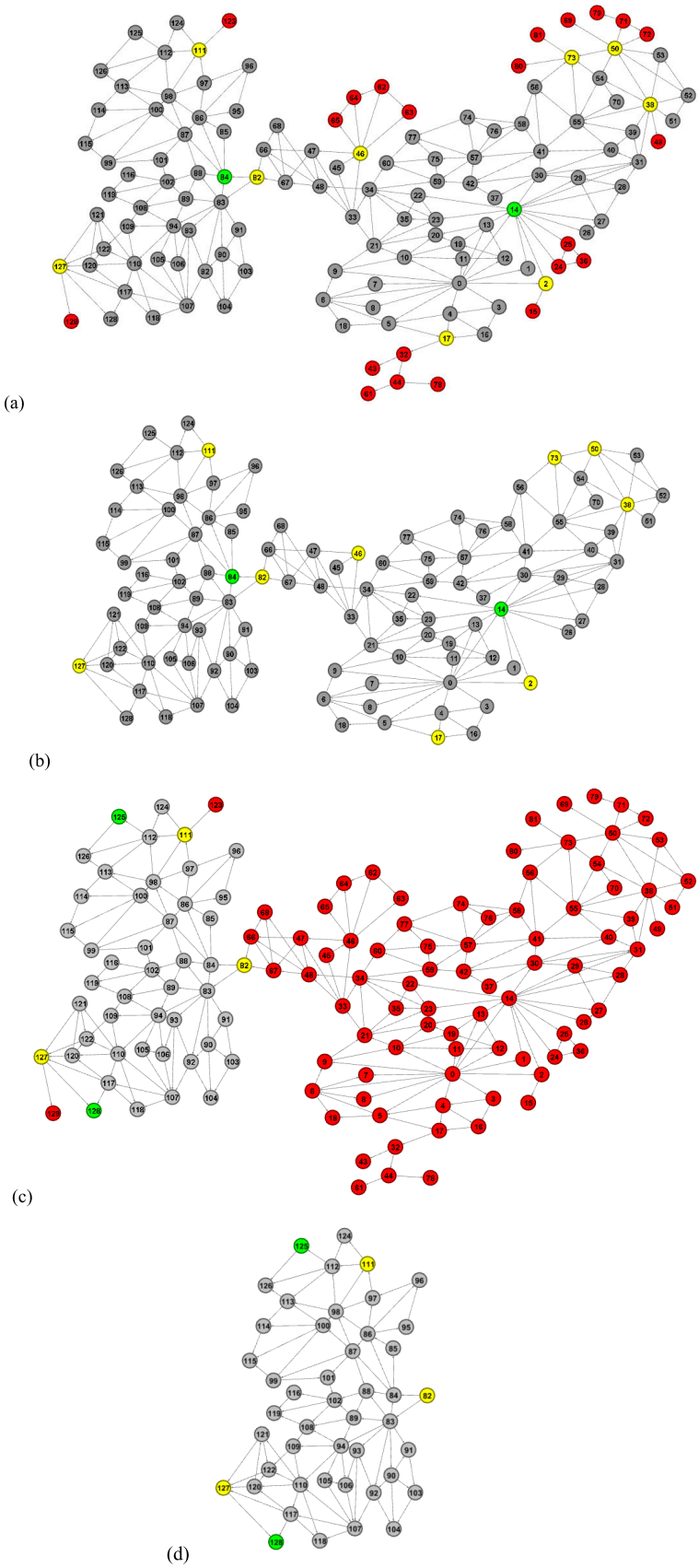


FIGURE 14. SCS Network simplification, (a) original SCS Network, $K = \{14, 84\}$, (b) valid subnetwork, $K = \{14, 84\}$, (c) original network, $K = \{125, 128\}$, (d) valid subnetwork, $K = \{125, 128\}$.

A. ORDERING HEURISTICS

The size of BDD (i.e., the number of non-sink nodes) encoding a network heavily depends on the chosen ordering of the network components. Empirical studies show that there is often a variation of several orders of magnitude between the sizes of BDD of the same network built using different orderings [31]. From a theoretical point of view, finding the best ordering for a relatively large Boolean reliability problem is an intractable task [32]; heuristics have been used to obtain good orderings.

The ordering heuristics commonly used to compute the network reliability are typically based on *breadth-first search* (BFS) or *depth-first search* (DFS) [31].

The DFS begins at a root vertex and explores as far as possible along each branch before backtracking. In the implementation, edge ordering with DFS uses a last-in-first-out stack data structure to store intermediate results as it traverses a network graph. To illustrate the DFS heuristic, a simple SCS network is used, and the DFS ordering results on all the edges using vertex 0 as the root vertex are given in Fig.12(a). Note that the successors of the currently processed vertex are chosen according to the increasing indices of the vertices during the DFS search process. For example, vertex 0 has 5 successor vertices 1, 2, 3, 6 and 7, vertex 1 is processed before the other vertices.

A key difference between BFS and DFS is the order in which the discovered (adjacent) vertices are explored. The BFS begins at a root vertex and inspects all its successor vertices. Then for each of these successor vertices in turn, it inspects their successor vertices that remain unvisited, and so on. In the implementation, edge ordering with BFS uses a first-in-first-out queue data structure to store intermediate results as it traverses a network graph. Fig. 12(b) illustrates the BFS ordering results for the network example in Fig.12(a) using vertex 0 as the root vertex. The successors of the currently processed vertex are processed in the order of their increasing indices.

B. BENCHMARK SCS ONE

The first benchmark SCS deployed in industry services [33]–[35] has totally 130 vertices and 276 edges as shown in Fig.13.

Consider $K = \{14, 84\}$, which are marked with green color in Fig.14(a). By applying the proposed network simplification method, we mark all articulation vertices with yellow color and all vertices in *RV* with red color. The simplified SCS is obtained by removing all vertices in *RV* and associated edges, as shown in Fig.14(b). It has 108 vertices and 248 edges.

Consider $K = \{125, 128\}$, which are marked with green color in Fig.14(c). By applying our simplification method, the simplified SCS is shown in Fig.13(d), which has 46 vertices and 102 edges.

Table 1 gives the experimental data showing the performance improvement achieved by our simplification method. Column 'T_1' gives the K-terminal network reliability evaluation time by applying the BDD-based algorithms to the original network. Column 'T_2' gives the running time of the network simplification. Column 'T_3' gives the K-terminal network reliability evaluation time by applying the BDD-based algorithm to the simplified network. The last column gives the improvement ratio in terms of $(T_1 - T_2 - T_3)/T_1$.

TABLE 1. Performance improvement of the proposed simplification method.

K	Running time (ms)			Improvement
	T_1	T_2	T_3	
{14,84}	32885.5	99.9	28335.5	13.53%
{125,128}	13121.1	85.1	10912.8	16.18%

Nest we investigate the following two questions:

Question 1: Does BFS heuristic still outperform the DFS heuristic for the SCS?

Question 2: How to improve the heuristic performance for SCS networks?

To answer **Question 1**, we test DFS and BFS heuristics on two SCS networks in Fig.14(b) and Fig.14(d). Table 2 gives the experimental data of performance comparison. The '*Rel*' column indicates the evaluated K-terminal reliability when each edge is subject to failure with a known probability 0.1. The '*No. ite*' column is the total number of *if-then-else* expressions generated during the BDD model construction. The '*BDDSize*' column is the total number of non-sink nodes in the final BDD model encoding a given K-terminal network. The '*Time*' column is total running time of K-terminal reliability evaluation. The columns '*Size_Comp.*' and '*Time_Comp.*' give the performance comparison in terms of *BDDSize*(*DFS*)/*BDDSize*(*BFS*) and *Time*(*DFS*)/*Time*(*BFS*).

From the results in Table 2, we get following conclusion, which does not conform to the widely accepted conclusion '*BFS heuristic outperforms the DFS heuristic*' drawn from the performance tests based on regular networks [31].

Conclusion 1: For SCS networks, the DFS heuristic is more likely to achieve better performance in BDD size (and also the running time) than the BFS heuristic.

Normally, the successors of the currently processed vertex are chosen according to the preassigned indices of the vertices during the DFS/BFS search process. This scheme has two disadvantages: 1) the same heuristic, applied to different vertex numbering methods of the same network, can give dramatically different results; 2) it gives little respect to the inhomogeneous degree distributions found in the SCS networks.

To answer **Question 2** (How to improve the heuristic performance for SCS networks?), we use an improvement technique called 'weighting' to design some variants based on BFS and DFS where a weight is computed and assigned to each vertex before the graph search starts. Then, the successors of the currently processed vertex are chosen according to the weights of the vertices. Here, we use the degree of a vertex as its weight. There are two classes of weighting ordering:

TABLE 2. Performance comparison: DFS vs. BFS.

	K	*DFS*				*BFS*				*Size_Comp.*	*Time_Comp.*
		Rel.	*No. ite*	*BDDSize*	*Time*	*Rel.*	*No. ite*	*BDDSize*	*Time*		
Net1 Fig.14(b)	{50,128}	0.999999	495886	2032	31.3	0.999999	2230579	413097	1094.3	<1%	3%
	{74,88}	0.998899	533749	2762	106.6	0.998899	2779033	442925	3797.1	1%	3%
	{30,114,5}	0.989998	575539	3098	114.4	0.989998	3369723	476148	2477.8	1%	5%
	{2,100,52}	0.989888	652741	6202	166.6	0.989888	3525440	68532	626.3	9%	27%
	{128,70}	0.890998	682060	3024	132.9	0.890998	3596802	61251	330.6	5%	40%
Net2 Fig.14(d)	{96,127}	0.999871	769136	4935	125.6	0.999871	3710270	33477	235.3	15%	53%
	{109,82,114}	0.999990	819471	3946	81.5	0.999990	3781421	23937	167.2	16%	49%
	{119,104,126}	0.989981	878939	4471	153.2	0.989981	3874142	24627	929.6	18%	16%
	{121,95}	0.989991	992891	5506	726.2	0.989991	4173279	162406	1058.4	3%	69%
	{103,115}	0.989979	1069729	11178	173.0	0.989979	4295031	40781	568.1	27%	30%

TABLE 3. Performance comparison: DFS vs. IDDFS vs. DDDFS vs. BFS vs. IDBFS vs. DDBFS.

	K	*BDDSzie*						*Comp*1.	*Comp*2.	*Comp*3.	*Comp*4.	*Comp*5.
		DFS	*IDDFS*	*DDDFS*	*BFS*	*IDBFS*	*DDBFS*					
Net1 Fig.14(b)	{50,128}	2032	14057	1354	413097	7265	258030	67%	10%	2%	3%	19%
	{74,88}	2762	22313	1827	442925	7146	268629	66%	8%	2%	3%	26%
	{30,114,5}	3098	22501	2448	476148	7980	365920	79%	11%	2%	2%	31%
	{2,100,52}	6202	45223	3891	68532	33113	79234	63%	9%	48%	42%	12%
	{128,70}	3024	48511	7126	61251	4336	67887	236%	15%	7%	6%	164%
Net2 Fig.14(d)	{96,127}	4935	18764	4875	33477	25979	49706	99%	26%	78%	52%	19%
	{109,82,114}	3946	13499	3084	23937	16016	70860	78%	23%	67%	23%	19%
	{119,104,126}	4471	22071	8050	24627	21303	74630	180%	36%	87%	29%	38%
	{121,95}	5506	6119	4365	162406	97409	181182	79%	71%	60%	54%	4%
	{103,115}	11178	28251	8621	40781	50986	362854	77%	31%	125%	14%	17%

increasing degree ordering and decreasing degree ordering. For the increasing degree ordering, the discovered (adjacent) vertices are explored in the order of their increasing degrees, whereas, they are explored in the order of their decreasing degrees for the decreasing degree ordering.

As an illustration, ordering results of the increasing degree DFS (IDDFS), decreasing degree DFS (DDDFS), increasing degree BFS (IDBFS), and decreasing degree BFS (DDBFS) are shown in Fig.12(c)-Fig.12(f).

Table 3 gives the results of performance comparison among ordinary DFS, ordinary BFS, IDDFS, DDDFS. IDBFS and DDBFS. The columns '*Comp*1.', '*Comp*2.', '*Comp*3.', '*Comp*4.', and '*Comp*5.' give the performance comparison in terms of *BDDSize*(*DDDFS*) / *BDDSize*(*DFS*), *BDDSize* (*DDDFS*) / *BDDSize* (*IDDFS*), *BDDSize*(*IDBFS*) / *BDDSize*(*BFS*), *BDDSize* (*IDBFS*) / *BDDSize* (*DDBFS*), and *BDDSize*(*DDDFS*) / *BDDSize*(*IDBFS*).

For DFS-style heuristics, it can be concluded that:

Conclusion 2: For SCS networks, the DDDFS heuristic is more likely to achieve better performance in BDD size than the other two DFS-style heuristics, i.e., choosing the successors of the currently processed vertex in the decreasing degrees order can achieve better performance with a high probability.

For BFS-style heuristics, we get the following conclusion, which is different from **Conclusion 2**drawn from the performance tests based on DFS.

Conclusion 3: For SCS networks, IDBFS heuristic is more likely to achieve better performance in BDD size than the other two BFS-style heuristics. Specifically, DDBFS heuristic has poor performance because for only one case it can outperform IDBFS but still worse than ordinary BFS.

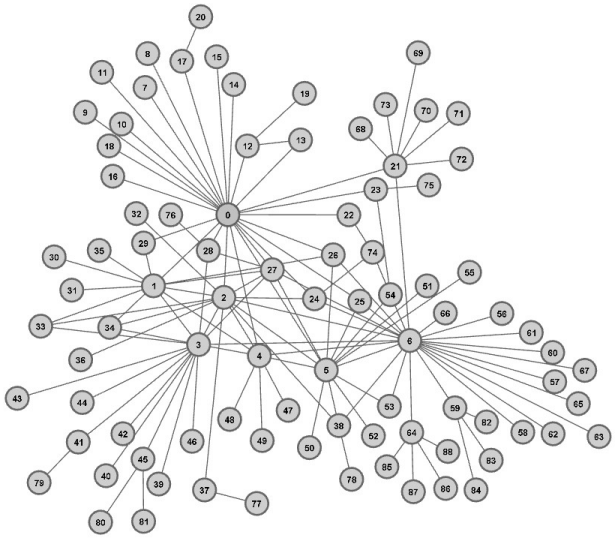


FIGURE 15. The second benchmark SCS deployed in a smart grid.

Finally, according to the comparison results between best DFS-style heuristic 'DDDFS' and best BFS-style 'IDBFS', It can be concluded that:

Conclusion 4: For SCS networks, DDDFS heuristic is more likely to achieve better performance in BDD size than IDBFS.

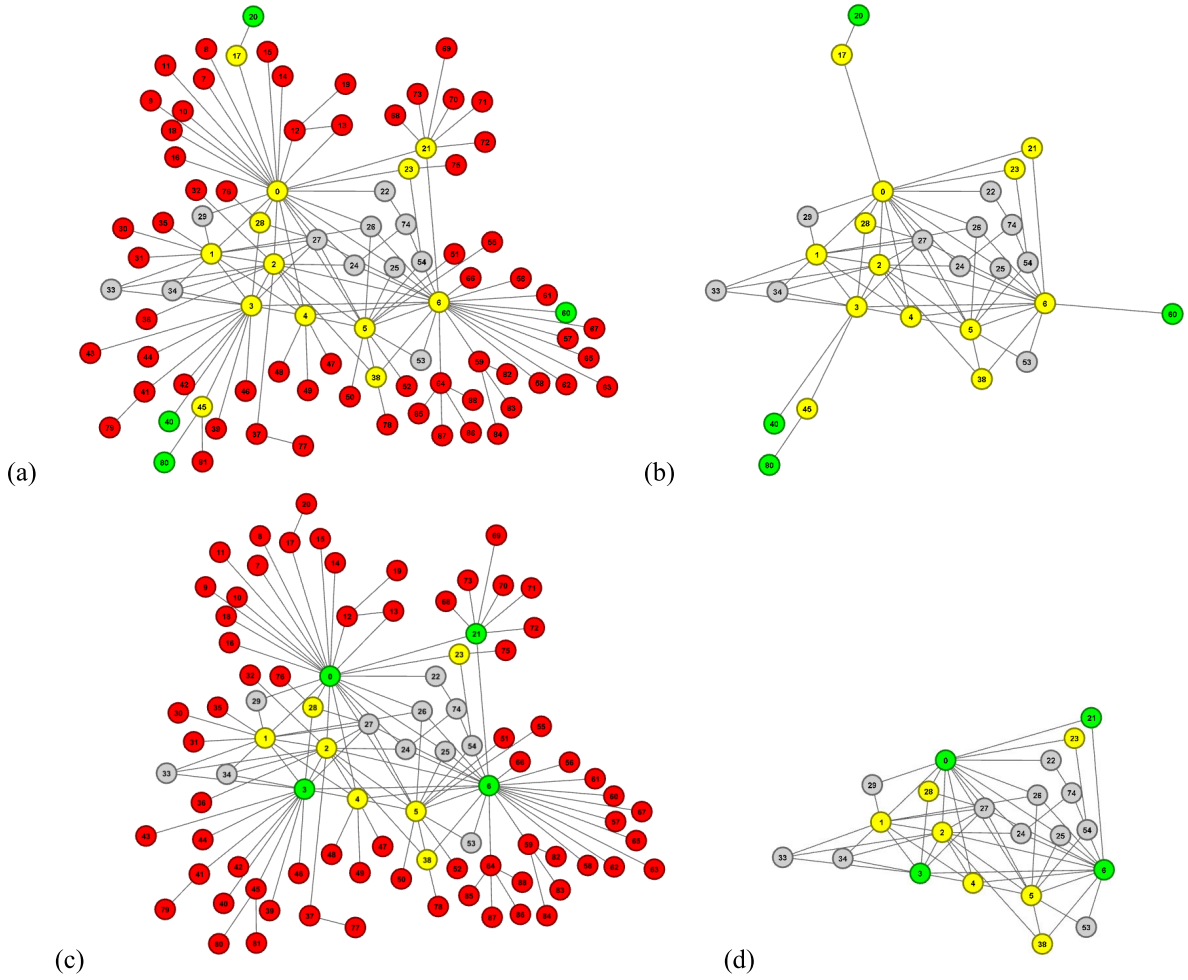


FIGURE 16. SCS simplification, (a) original network, $K = \{20, 40, 60, 80\}$, (b) valid subnetwork, $K = \{20, 40, 60, 80\}$, (c) original CN, $K = \{0, 3, 6, 21\}$, (d) valid subnetwork, $K = \{0, 3, 6, 21\}$.

TABLE 4. Performance comparison: DFS vs. BFS.

	K	*DFS*				*BFS*				*Size_Comp.*	*Time_Comp.*
		Rel.	*No. ite*	*BDDSize*	*Time*	*Rel.*	*No. ite*	*BDDSize*	*Time*		
Net1 Fig.16(b)	{0,45,60}	0.999845	1674	323	19.6	0.999845	3881	2371	22.9	14%	42%
	{21,33}	0.989928	3420	758	12.7	0.989928	4808	2944	26.6	26%	48%
	{22,4,80}	0.989872	2889	577	13.4	0.989872	5285	2976	27.1	19%	49%
	{27,53,1}	0.979974	4271	737	13.7	0.979974	8123	4342	32.0	17%	43%
	{26,29,5}	0.979100	2964	590	12.7	0.979100	3712	1524	25.9	39%	49%
Net2 Fig.16(d)	{21,33}	0.999992	929	154	20.0	0.999992	30371	21404	60.8	1%	33%
	{23,5,3}	0.997918	2112	292	22.1	0.997918	36160	30569	89.5	1%	25%
	{22,29,38}	0.989956	1666	242	26.7	0.989956	50633	36251	123.5	1%	22%
	{24,33,38}	0.987909	2795	470	27.7	0.987909	45456	24505	125.4	2%	22%
	{27,4,74}	0.980060	2550	460	29.5	0.980060	28555	15673	82.3	3%	36%

C. BENCHMARK SCS TWO

The second benchmark SCS deployed in a smart grid services [36, 37] has totally 89 vertices and 126 edges as shown in Fig.15.

Consider $K = \{20, 40, 60, 80\}$, which are marked with green color in Fig.16(a). By applying our simplification method, we mark all articulation vertices with yellow color and all vertices in *RV* with red color. With the removal of all vertices in *RV* and associated edges, the simplified SCS is shown in Fig.16(b), which has 28 vertices and 64 edges.

Consider $K = \{0, 3, 6, 21\}$, which are marked with green color in Fig.16(c). By applying our simplification method, the simplified SCS is shown in Fig.16(d), which has 22 vertices and 58 edges.

Table 4 gives the experimental data of performance comparison. The '*Rel*' column indicates the evaluated *K*-terminal

TABLE 5. Performance comparison: DFS vs. IDDFS vs. DDDFS vs. BFS vs. IDBFS vs. DDBFS.

	K	*BDDSzie*						*Comp*1.	*Comp*2.	*Comp*3.	*Comp*4.	*Comp*5.
		DFS	*IDDFS*	*DDDFS*	*BFS*	*IDBFS*	*DDBFS*					
Net1 Fig.16(b)	{0,45,60}	323	432	262	2371	893	4159	81%	61%	38%	21%	29%
	{21,33}	758	918	409	2944	1147	6502	54%	45%	39%	18%	36%
	{22,4,80}	577	755	260	2976	979	6257	45%	34%	33%	16%	27%
	{27,53,1}	737	809	510	4342	972	4219	69%	63%	22%	23%	52%
	{26,29,5}	590	397	769	1524	2456	5799	130%	194%	161%	42%	31%
Net2 Fig.16(d)	{21,33}	154	180	179	21404	6379	27939	116%	99%	30%	23%	3%
	{23,5,3}	292	327	289	30569	6848	40986	99%	88%	22%	17%	4%
	{22,29,38}	242	308	321	36251	10422	51921	133%	104%	29%	20%	3%
	{24,33,38}	470	461	216	24505	17874	90177	46%	47%	73%	20%	1%
	{27,4,74}	460	430	319	15673	13004	59611	69%	74%	83%	22%	2%

TABLE 6. Performance improvement of our simplification method.

K	Running time (ms)			Improvement
	T_1	T_2	T_3	
{20,40,60,80}	127.98	0.67	68.17	46.19%
{0,3,6,21}	107.72	0.68.9	63.95	40.00%

reliability when each edge is subject to failure with a known probability 0.1.

Table 5 gives the results of performance comparison among ordinary DFS, ordinary BFS, IDDFS, DDDFS. IDBFS and DDBFS.

Table 6 gives the experimental data illustrating the performance improvement achieved by our simplification method combined with the BDD-based algorithm. The data in Table 6 show that the proposed simplification method can significantly improve the efficiency of the BDD-based K-terminal network reliability evaluation.

V. CONCLUSION

In this article,, we propose a network simplification method that can effectively reduce a large network into a smaller valid subnetwork with significantly fewer edges and vertices while retaining the essential K-terminal communication structures for correct network reliability analysis [38], [39]. The method is based on graph decomposition and reconstruction through articulation vertices and has polynomial time complexity. Based on the generated valid subnetwork having fewer elements (edges and vertices), the BDD model can be constructed more quickly by avoiding inefficient manipulation of numerous redundant graph searching and *ite* expansions involved in the analysis based on the original network.

In this article,, we also investigate the high-performance edge ordering for SCS networks. For regular network benchmarks, the BFS heuristic was shown to outperform other heuristics like DFS in general and thus has been commonly used to generate the ordering for the BDD-based network reliability analysis. However, different conclusions have been derived for SCS networks: 1) the DFS heuristic outperforms the BFS heuristic for most of tested SCS networks; 2) choosing successors of the currently processed vertex according to their degrees may achieve better performance, and DFS and BFS heuristics have different preference for the degree ordering, particularly, the DFS heuristic performs better when choosing successors in the decreasing degrees order while the BFS heuristic performs better when choosing successors in the increasing degrees order.

This work has focused on the K-terminal network reliability, which is mainly concerned with the connectivity of SCSs. In the future we are interested in extending the proposed methodology to consider other characteristics (e.g., sensing coverage, network loading) that contribute to the reliability performance of SCSs [40].

With the drastic development of IoT, the number of connected mobile users is increasing at an unprecedented speed. The increasing popularity of mobile devices has triggered more and more new mobile applications. Thus, we are also interested in extending the K-terminal network reliability analysis for applications in the edge-cloud environment to help the design of effective offloading strategies for multiple mobile devices [41].

REFERENCES

[1] L. Xing, "Reliability in Internet of Things: Current status and future perspectives," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6704–6721, Aug. 2020, doi: 10.1109/JIOT.2020.2993216.

[2] S. Wan, Y. Zhao, T. Wang, Z. Gu, Q. H. Abbasi, and K.-K.-R. Choo, "Multi-dimensional data indexing and range query processing via Voronoi diagram for Internet of Things," *Future Gener. Comput. Syst.*, vol. 91, pp. 382–391, Feb. 2019.

[3] D. Leonard, Z. Yao, V. Rai, and D. Loguinov, "On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks," *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 644–656, Jun. 2007.

[4] Y. Mo, L. Xing, W. Guo, S. Cai, Z. Zhang, and J. Jiang, "Reliability analysis of IoT networks with community structures," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 304–315, Jan. 2020.

[5] M. O. Ball, "Computational complexity of network reliability analysis: An overview," *IEEE Trans. Rel.*, vol. 35, no. 3, pp. 230–239, 1986.

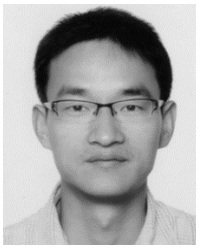
[6] W.-C. Yeh, "An improved sum-of-disjoint-products technique for the symbolic network reliability analysis with known minimal paths," *Rel. Eng. Syst. Saf.*, vol. 92, no. 2, pp. 260–268, Feb. 2007.

[7] Y. Kim and W.-H. Kang, "Network reliability analysis of complex systems using a non-simulation-based method," *Rel. Eng. Syst. Saf.*, vol. 110, pp. 80–88, Feb. 2013.

[8] W.-H. Kang, J. Song, and P. Gardoni, "Matrix-based system reliability method and applications to bridge networks," *Rel. Eng. Syst. Saf.*, vol. 93, no. 11, pp. 1584–1593, Nov. 2008.

[9] L. Chang and J. Song, "Matrix-based system reliability analysis of urban infrastructure networks: A case study of MLGW natural gas network," in *Proc. 5th China-Jpn.-US Trilateral Symp. Lifeline Earthquake Eng.*, 2007, pp. 21–29.

[10] G. Levitin, "Reliability evaluation for acyclic consecutively connected networks with multistate elements," *Rel. Eng. Syst. Saf.*, vol. 73, no. 2, pp. 137–143, Aug. 2001.
[11] W.-C. Yeh and Y.-M. Yeh, "A novel label universal generating function method for evaluating the One-to-all-Subsets general multistate information network reliability," *IEEE Trans. Rel.*, vol. 60, no. 2, pp. 470–478, Jun. 2011.
[12] W.-C. Yeh, "A modified universal generating function algorithm for the acyclic binary-state network reliability," *IEEE Trans. Rel.*, vol. 61, no. 3, pp. 702–709, Sep. 2012.
[13] K. Yan, C. Zhong, Z. Ji, and J. Huang, "Semi-supervised learning for early detection and diagnosis of various air handling unit faults," *Energy Buildings*, vol. 181, pp. 75–83, Dec. 2018.
[14] K. Yan, L. Ma, Y. Dai, W. Shen, Z. Ji, and D. Xie, "Cost-sensitive and sequential feature selection for chiller fault detection and diagnosis," *Int. J. Refrig.*, vol. 86, pp. 401–409, Feb. 2018.
[15] P. L'Ecuyer, G. Rubino, S. Saggadi, and B. Tuffin, "Approximate zero-variance importance sampling for static network reliability estimation," *IEEE Trans. Rel.*, vol. 60, no. 3, pp. 590–604, Sep. 2011.
[16] H. Cancela, P. L'Ecuyer, G. Rubino, and B. Tuffin, "Combination of conditional Monte Carlo and approximate zero-variance importance sampling for network reliability estimation," in *Proc. Winter Simulation Conf.*, Dec. 2010.
[17] Z. I. Botev, P. L'Ecuyer, G. Rubino, R. Simard, and B. Tuffin, "Static network reliability estimation via generalized splitting," *Informs J. Comput.*, vol. 25, no. 1, pp. 56–71, Feb. 2013.
[18] S. B. Akers, "Binary decision diagrams," *IEEE Trans. Comput.*, vol. C-27, no. 6, pp. 509–516, Jun. 1978.
[19] Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677–691, Aug. 1986.
[20] S. Li, S. Si, H. Dui, Z. Cai, and S. Sun, "A novel decision diagrams extension method," *Rel. Eng. Syst. Saf.*, vol. 126, pp. 107–115, Jun. 2014.
[21] R. Peng, H. Mo, M. Xie, and G. Levitin, "Optimal structure of multi-state systems with multi-fault coverage," *Rel. Eng. Syst. Saf.*, vol. 119, pp. 18–25, Nov. 2013.
[22] Y. Mo, "A multiple-valued decision-diagram-based approach to solve dynamic fault trees," *IEEE Trans. Rel.*, vol. 63, no. 1, pp. 81–93, Mar. 2014.
[23] Y. Mo, L. Xing, and S. V. Amari, "A multiple-valued decision diagram based method for efficient reliability analysis of non-repairable phased-mission systems," *IEEE Trans. Rel.*, vol. 63, no. 1, pp. 320–330, Mar. 2014.
[24] G. Levitin, L. Xing, S. V. Amari, and Y. Dai, "Reliability of non-repairable phased-mission systems with propagated failures," *Rel. Eng. Syst. Saf.*, vol. 119, pp. 218–228, Nov. 2013.
[25] S.-Y. Kuo, S.-K. Lu, and F.-M. Yeh, "Determining terminal-pair reliability based on edge expansion diagrams using OBDD," *IEEE Trans. Rel.*, vol. 48, no. 3, pp. 234–246, 1999.
[26] F.-M. Yeh, S.-K. Lu, and S.-Y. Kuo, "OBDD-based evaluation of k-terminal network reliability," *IEEE Trans. Rel.*, vol. 51, no. 4, pp. 443–451, Dec. 2002.
[27] S.-Y. Kuo, F.-M. Yeh, and H.-Y. Lin, "Efficient and exact reliability evaluation for networks with imperfect vertices," *IEEE Trans. Rel.*, vol. 56, no. 2, pp. 288–300, Jun. 2007.
[28] G. Hardy, C. Lucet, and N. Limnios, "K-terminal network reliability measures with binary decision diagrams," *IEEE Trans. Rel.*, vol. 56, no. 3, pp. 506–515, Sep. 2007.
[29] X. Zang, H. Sun, and K. Trivedi, "A BDD-based algorithm for reliability graph analysis," Dept. Elect. Eng., Duke Univ., Durham, NC, USA, Tech. Rep. DU-CACC-9418764, 2000.
[30] J. Hopcroft and R. Tarjan, "Algorithm 447: Efficient algorithms for graph manipulation," *Commun. ACM*, vol. 16, no. 6, pp. 372–378, Jun. 1973.
[31] Y. Mo, L. Xing, F. Zhong, Z. Pan, and Z. Chen, "Choosing a heuristic and root node for edge ordering in BDD-based network reliability analysis," *Rel. Eng. Syst. Saf.*, vol. 131, pp. 83–93, Nov. 2014.
[32] S. J. Friedman and K. J. Supowit, "Finding the optimal variable ordering for binary decision diagrams," in *Proc. 24th ACM/IEEE Conf. Design Autom. Conf. (DAC)*, 1987, pp. 348–356.
[33] C. K. Wu, K. F. Tsang, Y. Liu, H. Zhu, H. Wang, and Y. Wei, "Critical Internet of Things: An interworking solution to improve service reliability," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 74–79, Jan. 2020.
[34] C. Wang, L. Xing, V. M. Vokkarane, and Y. L. Sun, "Infrastructure communication sensitivity analysis of wireless sensor networks," *Qual. Rel. Eng. Int.*, vol. 32, no. 2, pp. 581–594, Mar. 2016.
[35] Y. Zeng, Y. Sun, L. Xing, and V. Vokkarane, "A study of online social network privacy via the TAPE framework," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 7, pp. 1270–1284, Oct. 2015.
[36] K. Yan, X. Wang, Y. Du, N. Jin, H. Huang, and H. Zhou, "Multi-step short-term power consumption forecasting with a hybrid deep learning strategy," *Energies*, vol. 11, no. 11, p. 3089, Nov. 2018.
[37] K. Yan, W. Li, Z. Ji, M. Qi, and Y. Du, "A hybrid LSTM neural network for energy consumption forecasting of individual households," *IEEE Access*, vol. 7, pp. 157633–157642, 2019.
[38] L. Xing and S. V. Amari, *Binary Decision Diagrams and Extensions for System Reliability Analysis*. Hoboken, NJ, USA: Wiley, Jul. 2015.
[39] L. Xing, G. Levitin, and C. Wang, *Dynamic System Reliability: Modeling and Analysis of Dynamic and Dependent Behaviors*. Hoboken, NJ, USA: Wiley, Mar. 2019.
[40] L. Xing and G. Levitin, "BDD-based reliability evaluation of phased-mission systems with internal/external common-cause failures," *Rel. Eng. Syst. Saf.*, vol. 112, pp. 145–153, Apr. 2013.
[41] K. Peng, H. Huang, S. Wan, and V. C. M. Leung, "End-edge-cloud collaborative computation offloading for multiple mobile users in heterogeneous edge-server environment," *Wireless Netw.*, to be published, doi: 10.1007/s11276-020-02385-1.



YUCHANG MO (Senior Member, IEEE) was a Visiting Scholar with the Department of Electrical and Computer Engineering, University of Massachusetts (UMass) Dartmouth, USA, from September 2012 to August 2013. He is currently a Distinguished Professor with the School of Mathematical Sciences, Huaqiao University, Quanzhou, China. His current research interest includes reliability analysis of complex systems and networks using combinatorial models. His research has been supported by the National Science Foundation of China.

MIN LIANG received the bachelor's degree from the Tianjin University of Commerce. She is currently pursuing the master's degree with Huaqiao University. Her research interest includes data science.

LIUDONG XING (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Virginia, in 2002.

She is currently a Professor with the Department of Electrical and Computer Engineering, University of Massachusetts (UMass) Dartmouth, USA. She has published two books titled *Binary Decision Diagrams and Extensions for System Reliability Analysis* and *Dynamic System Reliability: Modeling and Analysis of Dynamic and Dependent Behaviors*. Her current research interests include reliability and resilience modeling, analysis, and optimization of complex systems and networks. She is a Fellow of the International Society of Engineering Asset Management. She was a recipient of the 2014 Leo M. Sullivan Teacher of the Year Award, the 2010 Scholar of the Year Award, and the 2011 Outstanding Women Award of UMass Dartmouth. She was a recipient of the 2018 IEEE Region 1 Outstanding Teaching in an IEEE Area of Interest (University or College) Award, the 2015 ChangJiang Scholar Award, and the 2007 IEEE Region 1 Technological Innovation (Academic) Award. She was a co-recipient of the Best (Student) Paper Award at several conferences and journals. She is an Associate Editor or an Editorial Board member of multiple journals, including *Reliability Engineering and System Safety*, the IEEE Internet of Things Journal, and the *International Journal of Systems Science*.

JINPING LIAO received the bachelor's degree from Shangrao Normal University. She is currently pursuing the master's degree with Huaqiao University, China. Her research interests include artificial intelligence and machine learning.

XULI LIU received the bachelor's degree from the College of Science and Technology, Gannan Normal University. She is currently pursuing the master's degree with Huaqiao University, China. Her research interests include artificial intelligence and machine learning.