IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 16, 2019, accepted October 28, 2019, date of publication November 12, 2019, date of current version December 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2953180

A New Node-Based Concept for Solving the Minimal Path Problem in General Networks

WENBO ZHU[1], WEI-CHANG YEH[2], (Senior Member, IEEE), NEAL NAIXUE XIONG[3], AND BIN SUN[4]

[1]College of Automation, Foshan University, Foshan 528000, China
[2]Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 30013, Taiwan
[3]Department of Mathematics and Computer Science, Northeastern State University, Tahlequah, OK 74464, USA
[4]School of Aeronautics and Astronautics, University of Electronic Science and Technology of China, Chengdu 610051, China

Corresponding author: Wei-Chang Yeh (yeh@ieee.org)

This work was supported in part by the National Science Council of Taiwan under Grant NSC102-2221-E-007-086-MY3, and in part by the Open Project of State Key Lab of Rail Traffic Control & Safety under Grant RCS2019K010.

ABSTRACT The validation and evaluation of network reliability are significant issues in the planning, design, and control of systems. The minimal path (MP, an edge set) set is one of the main tools used in measuring network reliability. A new node-based concept to find all MPs is proposed by simply searching all of the ordered node subsets with a time complexity $O(ne^n)$, where n is the number of nodes. The proposed concept is implemented in a newly proposed node-based MP algorithm using depth-first search (DFS). Because $O(m) = O(n^2)$, where m is the number of edges, the proposed node-based MP algorithm is theoretically more efficient than conventional edge-based algorithms, which all use time complexity $O(n2^m)$ to search for all MPs. Finally, a series of numerical experiments is implemented to test the performance of the proposed algorithm.

INDEX TERMS General networks, reliability, minimal path (MP), node-based concept.

I. INTRODUCTION

Network reliability has been a research topic of interest since the early days of reliability engineering, and it has been a popular tool to evaluate and validate the performance of many real-world modern systems [1]–[40] such as grid and cloud computing [9], computer and communication systems [10], sensor networks [11], [12], traffic service networks [13], [14], the Internet of Things [15], [16], system design [17]–[19], and production systems [20].

The definition of (two-terminal) network reliability R is the success probability that there is at least an MP from the source node to the sink node, i.e., the probability for a live connection between a source node and a sink node in a network $G(V, E)$ [1]–[3], where V and E are the node set and edge set:

$$R = \Pr(\{X | \text{source node is connected to sink node in subnetwork } G(V, X) \text{ for all } X \subseteq E\}).$$

The network reliability problem is an NP-hard problem. There are different methods used for calculating network reliability, which are categorized into two types: exact algorithms and approximated algorithms. Approximated algorithms, e.g., the Monte-Carlo simulation method [18], [37]–[39], the reliability bounds [41], [42], and the Petri-net method [40], [40], attempt to determine the approximated reliability to reduce the computational burden. However, newer exact algorithms continually provide better results, as more powerful CPUs are able to accelerate their run-time, and the risk management of expensive and important systems is constantly improving; thus, determining the exact network reliability of medium-size networks has become somewhat easier [43]–[48].

Consequently, it is more helpful to approximate network reliability in managing and evaluating network reliability [1]–[15], [17], [19], [20], [35], [43]–[48]. This study thus only focuses on determining exact network reliability.

Most network reliability evaluation methods are judged in terms of minimal paths (MPs), which are edge subsets where the remaining set is no longer an MP if any edge is removed [43]–[47]. For example, $\{e_{12}, e_{25}, e_{54}, e_{46}\}$ is an MP from nodes 1 to 6 in Figure 1, where e_{ij} is a directed edge from nodes i to j.

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

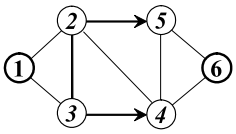

FIGURE 1. An example network.

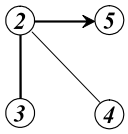

FIGURE 2. An example network to explain V2.

After finding all MPs, R can be easily calculated using the following equation:

$$R = \Pr(\{X | \text{there is at least an MP from the source node to the sink node in subnetwork} G(V, X) \text{ for all } X \subseteq E)\}.$$

However, it is also NP-hard to search for all MPs in general networks [43]–[47]. A variety of methods have been proposed to search for the entire MP set, e.g., the traditional depth-first-search method [43]–[45], the heuristic algorithm [46], the universal generating function method [47], etc. These approaches are edge-based and find all MPs by trying possible (unordered) edge subsets. In addition, each edge-based algorithm must generate and verify no more than 2^m MP candidates (edge subsets) with time complexity $O(n2^m)$, where n and m are the numbers of nodes and edges in the network [43]–[47].

Recently, Yeh proposed the first node-based MP algorithm to search for all MPs in a directed acyclic network by finding the longest path in each selected node subset [43]. The time complexity in Yeh's algorithm is $O(m2n^m)$, which is the first instance of the time complexity of the MP algorithm being equal to that of the best-known minimal cut (MC) algorithm (also node-based) in the literature [48], [49]. However, the directed acyclic network is only applicable for some special networks without any cycles, and it is also an NP-hard problem to find the longest path [43], i.e., Yeh's method is not applicable for general networks where cycles are always included. Thus, there is a need to develop a new node-based MP algorithm to improve the efficiency of the conventional edge-based MP algorithm for general networks.

The purpose of this paper is to propose a novel, straightforward, and simple node-based concept for the MP problem before calculating the binary-state network reliability. According to the proposed novel node-based concept, a novel node-based MP algorithm is developed to enumerate all MPs efficiently without needing to try all ordered node subsets. The computational complexity of the proposed node-based MP algorithm is analyzed to show its theoretical efficiency. In addition, to examine the practical performance of the proposed algorithm, two experiments are implemented. One compares the proposed node-based MP algorithm with the existing edge-based MP algorithm [43], [44] on 20 benchmark networks, and another tests the maximal size of complete networks that can be solved using the proposed algorithm. Note that there is always an edge between each pair of nodes in complete networks such that the number of edges of complete networks is maximum among all networks with the same number of nodes. The proposed node-based MP algorithm is also easily stretched to search for all MPs between all node pairs.

This paper is organized as follows. Section II describes the required acronyms, notations, nomenclature and assumptions. Section III presents the novel node-based concept for the MP problem. Its theoretical efficiency is also discussed in Section III by comparing the time complexity between the proposed node-based MP algorithm and the conventional edge-based MP algorithms. The major parts of the proposed MP algorithm based on the novel node-based concept are described in Section IV. Section V proposes the node-based MP algorithm with the depth-first-search strategy between two specific nodes in detail. In Section V, the proposed algorithm is illustrated step-by-step to show how to search for all MPs in networks. Two computational experiments are given in Section VI to compare the efficiency of the proposed algorithm with the conventional edge-based MP algorithms on 20 benchmark networks and evaluate the performance of the proposed algorithm by testing complete graphs on up to 16 nodes. Concluding remarks are given in Section VII.

II. NOTATIONS, NOMENCLATURE AND ASSUMPTIONS

A. ACRONYMS

MP: Minimal path.
MPN: The ordered node subset for the related MP.
DFS: The depth-first search.

B. NOTATIONS

$|\bullet|$: the number of elements of $\bullet$; e.g., $|E|$ is the number of nodes in E.
$G(V, E)$: a connected network, where $V=\{1, 2, 3, \ldots, n\}$ is the node set, E is the edge set, and nodes 1 and n are the specified source node and sink node, respectively; e.g., Figure 1 shows a network.
e_{ij}: $e_{ij} \in E$ is a directed edge from nodes i to j.
V_k: The adjacent node subset of node k $V_k = \{v_1, v_2, \ldots, v_\kappa\} \subseteq V$, where $e_{uv} \in E$ for all $u \in V_k$ and $i < j$ if $v_i < v_j$. Note that each adjacent node subset must be written in an ordered node subset in this study. For example, $V_2 = \{3, 4, 5\}$ in Figure 1 (see Figure 2).
$V_{k,i}$: The ith node in V_k. For example, $V_{2,1} = 3$ and $V_{2,3} = 5$ in $V_2 = \{3, 4, 5\}$.
$p(i)$: The ith node in the ordered node subset p. For example, $p(3) = 4$ and $p(5) = 6$ if $p = \{1, 2, 4, 5, 6\}$ in Figure 1.
$V^*(P)$: The ordered node subset such that MP P is from the first node to the last node via all nodes in $V^*(P)$. For example, in Figure 1, $V^*(P)=\{1, 2, 4,$

5, 6} is an MPN because $P = \{e_{12}, e_{25}, e_{54}, e_{46}\}$ is an MP from nodes 1 to 2 to 4 to 5 to 6 in V^*.

C. NOMENCLATURE

Reliability: the probability of a live connection between the source node and the sink node.

MP/MC: an arc subset that forms a path/cut such that if any arc is removed from this subset, then the remaining subset is no longer a path/cut.

MPN: an ordered node subset; the set of these arcs connected any two consecutive nodes in an MPN is an MP. For example, {1, 2, 5, 4, 6} is an MPN, and $\{e_{12}, e_{25}, e_{54}, e_{46}\}$ is the corresponding MP from nodes 1 to 6 in Figure 1.

Complete network (graph): a fully connected and undirected network such that there is always a distinctive arc between each pair of nodes; i.e., the number of its arcs is equal to $|V|(|V|-1)/2$.

D. ASSUMPTIONS

The network satisfies the following assumptions [43]–[47]:

1. Each node is perfectly reliable.
2. The graph is connected, without parallel edges, and free of self-loops.
3. Each edge has two states: working or failed.

III. THE NOVEL NODE-BASED CONCEPT

A novel node-based concept is presented in this section. In Section 5, it is implemented in the proposed node-based MP algorithm to enumerate all MPs with the help of the adjacent node subsets, index Lab(•), and index Pos(•) discussed in Section 4.

The following property and corollaries, which are simply based on the above definition of an MPN, discuss the relationships between the MP and the MPN.

Property 1: P is an MP if and only if $V^*(P)$ is an MPN.

Corollary 1: All MPs can be found in MPNs.

Corollary 2: All MPNs can be found from all ordered node subsets.

The next property, following Property 1 and Corollaries 1 and 2 directly, describes a novel concept to obtain the complete MP set by simply searching for all MPNs from all ordered node subsets.

Property 2: All MPs can be found from all ordered node subsets.

The following property shows the time complexity in implementing Property 2.

Property 3: It takes $O(ne^n)$ time to find all MPs when implementing Property 2.

Proof: The formula for the number of possible permutations of $(n-k)$ nodes from n nodes is $\frac{n!}{k!}$. Hence, the number of all possible ordered node subsets is $\sum_{k=1}^{n} \frac{n!}{k!}$. Moreover,

$$\frac{n!}{k!} < \frac{n^k}{k!}$$

$$\Rightarrow \sum_{k=1}^{n} \frac{n!}{k!} < \sum_{k=1}^{n} \frac{n^k}{k!}$$

$$\Rightarrow \sum_{k=1}^{n} \frac{n!}{k!} < \sum_{k=1}^{n} \frac{n^k}{k!} < \sum_{k=0}^{\infty} \frac{n^k}{k!} = e^n$$

$$\Rightarrow \sum_{k=1}^{n} \frac{n!}{k!} = \sum_{k=1}^{n} \frac{n!}{(n-k)!} < e^n.$$

In addition, the time complexity is $O(n)$ to verify whether each ordered node subset is an MP. Hence, it takes $O(ne^n)$ time to find all MPs by trying all ordered node subsets.

The following property states that there is no need to eliminate identical MPs after generating all MPs from MPNs.

Property 4: If q_1, q_2 are two distinctive MPNs, then the MP generated from q_1 is different from that of q_2.

Proof: Because $q_1 \neq q_2$ and all MPs from nodes 1 to n must be via all nodes in the related node subset.

Therefore, we have the following important theorem.

Theorem 1: All MPs can be found without any duplications from all ordered node subsets in time complexity $O(ne^n)$.

IV. THE DFS, ADJACENT NODE SUBSETS, INDEX Lab, AND INDEX Pos

If all ordered node subsets must be located to search for all MPs (MPNs) while the proposed node-based concept is implemented using Theorem 1, then this task becomes very time-consuming. Hence, a novel algorithm, called the node-based MP algorithm, is proposed to implement the novel node-based concept efficiently without trying all ordered node subsets.

The proposed algorithm adapts a depth-first search (DFS) to construct the search tree, called the DFS-tree here, to save memory space in efficiently enumerating MPNs [43-45]. A node is called a visited node if it is in the current path; otherwise, it is called an unvisited node in the DFS-tree. Let the node in the last position of the current path be a parent node, the node appended immediately after the parent node be an offspring node, and any other node qualified to be the offspring node be a brother node of that offspring node.

In a conventional DFS, the DFS-tree adopted in the algorithm starts at the source node with the following three steps for each branch during the search:

1) The offspring-branching step is as follows:
   a) Select and add an unvisited offspring node to the last node in the current path.
   b) If no unvisited node can be chosen in this step, proceed to the brother-branching step.
   c) If the offspring node is the sink node, then an MPN is found; save all ancestors in sequence (which is the found MPN) and proceed to the parent-branching step.
2) The brother-branching step is as follows:
   a) Replace the last node in the current path with one of its unvisited brother nodes and return to the offspring-branching step.

b) If no unvisited brother node can be chosen in this step, proceed to the parent-branching step.

3) The parent-branching step is as follows:

a) Return to the parent node of the current node and proceed to the brother-branching step.

b) If there is no parent node—i.e., the current node is the source node—halt.

To speed up the procedure in building the DFS tree, three innovations are proposed: the adjacent node subset, the index Lab(•), and the index Pos(•), where • is a node. Lab(•) is used to prevent any path found in the proposed algorithm with redundant edges or cycles; i.e., only the MPN is found by the proposed algorithm. Pos(•) is implemented to ascertain that each MP is found efficiently without duplications in the proposed algorithm.

The details of these three innovations are explained below.

A. THE ADJACENT NODE SUBSET

The following property shows that only the nodes adjacent to the parent node—say, v—can be the offspring node of node v.

*Property 5:*Let p be an ordered node subset, and let there always be an edge from nodes $p(i)$ to $p(i+1)$ in E for $i = 1, 2, \ldots, k-1$. Then, $p(i+1) \in V_{p(i)}$ for $i = 1, 2, \ldots, k-1$.

Proof: It follows from the definition of the adjacent node subset.

Property 5 is able to reduce the number of offspring node candidates and is also the core of indices Lab(•) and Pos(•). Note that the adjacent node subset of each node is built only in the initial step of the proposed algorithm, and these subsets never change their elements.

B. THE INDEX Lab(•)

The basic setting of Lab(•) is very simple:

$$\mathrm{Lab}(v) = \begin{cases} 0 & \text{if node } v \text{ is visited in the current} \\ & \text{path of the DFS-tree} \\ 1 & \text{otherwise} \end{cases}. \quad (1)$$

The value of Lab(v) is updated from 1 to 0 immediately if v is no longer included in the current path, regardless of whether v will be included again in another distinctive path later. Conversely, the value of Lab(v) must change from 0 to 1 as long as v is included in the current path.

With the help of Lab(•), all paths are able to avoid the occurrence of cycles. The following discusses how a cycle forms by adding a new edge in a path. This is necessary to prevent a cycle in paths and is often used in this subsection.

*Property 6:*The path from nodes $p(1)$ to $p(2)$ to $\ldots$to $p(k)$ includes a cycle if and only if $e_{p(i),p(i+1)} \in E$ and $p(i) = p(j)$, where p is an ordered node subset, $j \neq i$, and $i = 1, 2, \ldots,$ k-1.

The following property, directly from Properties 1, 2, 5 and 6, explores some special characteristics about ordered node subsets, MPs, MPNs, and/or paths.

*Property 7:*Let p be the ordered node subset of a path in the DFS-tree such that $e_{p(i),p(i+1)} \in E$ for $i = 1,2,\ldots,k-1$, where $k = |p|$. Then, all statements below are true.

a. $p(i+1) \in V_{p(i)}$ for $i = 1,2,\ldots,k-1$.

b. There is a path from nodes $p(1)$ to $p(2)$ to $p(3)$, $\ldots$, to $p(k)$.

c. If $e_{p(k),v} \in E$ and $v \neq p(i)$ for $i = 1,2,\ldots,k$, then there is a path from nodes $p(1)$ to $p(2)$, $\ldots$, to $p(k)$ to v.

d. If $e_{p(k),v} \in E$ and $v = p(i)$ for $i = 1,2,\ldots,k$, then there is a cycle in the path from nodes $p(1)$ to $p(2)$, $\ldots$, to $p(k)$ to v.

e. p is an MPN if and only if $p(1)=1$, $p(k) = n$, and $p(i) \neq p(j)$ for $i \neq j$.

The following property restates Property 7 in terms of the label function Lab(•). Property 8 also shows that all MPNs can be found by enumerating these ordered node subsets that form a path without redundant edges and cycles with the help of the label function Lab(•).

*Property 8:*Let p be an ordered node subset, $k = |p|$, $p(i+1) \in V_{p(i)}$ for $i = 1,2,\ldots,k-1$ and

$$\mathrm{Lab}(v) = \begin{cases} 0 & \text{if } v \notin p \\ 1 & \text{otherwise} \end{cases}.$$

Then, all statements below are true.

a. There is a path from nodes $p(1)$ to $p(2)$, $\ldots$, to $p(k)$ to v if $v \in V_{p(k)}$ and Lab(v)=0.

b. There is a cycle in the path from nodes $p(1)$ to $P(2)$, $\ldots$, to $p(k)$ to v if $v \in V_{p(k)}$ and Lab(v)=1.

c. No MP includes the subpath from nodes $p(1)$ to $p(2)$, $\ldots$, to $p(k)$ if $V_{p(k)} = \emptyset$.

C. THE INDEX Pos(•)

It is trivial that the DFS strategy is able to find all possible ordered node subsets by trying all paths without duplications [43]–[47]. However, it is burdensome and inefficient to find all possible ordered node subsets and then filter out all MPs from these ordered node subsets. Hence, another index called Pos(•) is proposed to reduce the solution space.

The following property shows that these ordered node subsets of any found and non-abandoned path in the DFS tree is an MPN; i.e., there is no need to verify the feasibility of all final ordered node subsets obtained in the DFS-tree.

Property 9: If p is an ordered node subset found in the DFS-tree, then p includes nodes 1 and n, and p is an MPN.

Proof: It is trivial that each ordered node subset starts from node 1 in the proposed DFS-tree. If any ordered node subset is terminated at node k and $k \neq n$ in the DFS-tree, then $\mathrm{Pos}(V_k) > |V_k|$, and node k is abandoned and moves back to its parent node. Thus, any found and non-abandoned path in the DFS tree included nodes 1 and n. In addition, there is a path from $p(1)$ to $p(2)$ to $\ldots$to $p(k) = n$ with no cycles, and $p(i) \neq p(j)$ for $i \neq j$ and $i, j \leq k$ from Section 4.2. Thus, p is an MPN.

The following property shows that each MPN is found in the DFS-tree if Pos(•) is used; i.e., there is no need to try all possible ordered node subsets.

Property 10: If p is an MPN, then p can be found in the DFS-tree using the index Pos(•).

Proof: Let $p(1)=1, p(2), \ldots, p(k)$ be found in the proposed DFS-tree but $p(1)=1, p(2), \ldots, p(k), p(k+1)$ be not in the DFS-tree, where $k+1 < |p|$. It is trivial that $p(k+1) \in V_{p(k)}$ from Property 5; i.e., $V_{p(k)} \neq \emptyset$. If $\text{Pos}(p(k)) > |V_{p(k)}|$, then either $p(k+1) = p(i)$ for some $i < k+1$ or $V_{p(k+1)} = \emptyset$. However, the former case results in a cycle that is impossible in the DFS-tree; the latter case causes p to not be an MPN because it cannot reach node n. Hence, $p(1)=1, p(2), \ldots, p(k), p(k+1)$ is also in the DFS-tree. In the same way, $p(1)=1, p(2), \ldots, p(k), p(k+1), \ldots, p(|p|) = n$ must be found in the DFS-tree; i.e., Property 10 is true.

The following property, derived from the characteristic of DFS [43]–[47], directly explains that no identical MPN is found in developing the DFS-tree; i.e., it is unnecessary to verify and remove the MPN duplications.

Property 11: There is no duplicate ordered node subset found in the DFS tree using the index Pos(•).

V. THE PROPOSED ALGORITHM AND EXAMPLE

This section first outlines the pseudocode of the proposed node-based MP algorithm and then demonstrates the proposed algorithm in a stepwise manner using a medium-sized network, as shown in Figure 1.

A. THE PSEUDOCODE OF THE PROPOSED NODE-BASED MP ALGORITHM

The details of the proposed node-based MP algorithm are described in the following steps for finding all MPs in each network between two specific nodes:

Algorithm: Find all MPs in a network $G(V, E)$.

Input: A connected graph $G(V, E)$ with node set V, edge set E, a source node (node 1), and a sink node (node n).

Output: The MP set in $G(V, E)$.

STEP 0. Find V_k, and let $\text{Lab}(k)=0$, $\text{Pos}(V_k)=1$, where $k=1, 2, \ldots, n$, and $\text{Lab}(1)= i = v = v_1=1$.

STEP 1. If $\text{Lab}(V_{v,\text{Pos}(V_v)})=0$, let $\text{Lab}(V_{v,\text{Pos}(V_v)})=1$, $i = i+1$, $v = v_i$, and $\text{Pos}(V_v)=1$. Otherwise, let $\text{Pos}(V_v) = \text{Pos}(V_v)+1$ and go to STEP 4.

STEP 2. If $V_{v,\text{Pos}(V_v)}$ is not the sink node, go to STEP 1. Otherwise, a new MPN $\{v_1, v_2, \ldots, v_i\}$ is found and stored.

STEP 3. If $i=0$, then halt. Otherwise, let $\text{Lab}(v_i)=0$, $i = i-1$, and $v = v_i$.

STEP 4. If $\text{Pos}(V_v) = |V_v|$, go to STEP 3. Otherwise, let $\text{Pos}(V_v)=\text{Pos}(V_v)+1$ and go to STEP 1.

The correctness and time complexity of the proposed algorithm are based on Sections 3 and 4.

Theorem 1: The above algorithm searches for all MPs without duplicates with the time complexity $O(ne^n)$.

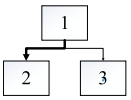

FIGURE 3. The DFS-tree with offspring node 2 of Figure 1.

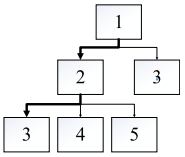

FIGURE 4. The DFS-tree with offspring node 3 of Figure 1.

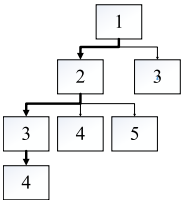

FIGURE 5. The DFS-tree with offspring node 4 of Figure 1.

B. AN EXAMPLE

For convenience and ease of understanding the proposed algorithm, Figure 1 is selected to demonstrate the general step-by-step procedure of the proposed node-based MP algorithm to search for all MPs between the source node (i.e., node 1) and the sink node (i.e., node 6) as follows:

Solution:

STEP 0. Let $V_1=\{2, 3\}$, $V_2=\{3, 4, 5\}$, $V_3=\{2, 4\}$, $V_4=\{2, 5, 6\}$, $V_5=\{4, 6\}$, $\text{Lab}(k)=0$, $\text{Pos}(V_k)=1$ for $k=1, \ldots, 6$, and $\text{Lab}(1)= i = v = v_1=1$.

STEP 1. Because $\text{Lab}(V_{v,\text{Pos}(V_v)})=\text{Lab}(V_{1,1})=\text{Lab}(2)=0$, let $\text{Lab}(2)=1$, $i = i+1=2$, $v = v_i = v_2=2$ (see Fig. 3), and $\text{Pos}(V_v) = \text{Pos}(V_2) = 1$.

STEP 2. Because node 2 is not the sink node, go to STEP 1.

STEP 1. Because $\text{Lab}(V_{v,\text{Pos}(V_v)}) = \text{Lab}(V_{2,1}) = \text{Lab}(3)=0$, let $\text{Lab}(3)=1$, $i = i+1=3$, $v = v_i = v_3 =3$ (see Fig. 4), and $\text{Pos}(V_v) = \text{Pos}(V_3) = 1$.

STEP 2. Because node 3 is not the sink node, go to STEP 1.

STEP 1. Because $\text{Lab}(V_{v,\text{Pos}(V_v)}) = \text{Lab}(V_{3,1}) = \text{Lab}(2)=0$, let $\text{Pos}(V_v) = \text{Pos}(V_v)+1=2$ and go to STEP 4.

STEP 4. Because $\text{Pos}(V_v) = |V_v| =2$, go to STEP 1.

STEP 1. Because $\text{Lab}(V_{v,\text{Pos}(V_v)}) = \text{Lab}(V_{3,2}) = \text{Lab}(4)=0$, let $\text{Lab}(4)=1$, $i = i+1=4$, $v = v_i = v_4 =4$ (see Fig. 5), and $\text{Pos}(V_v) =1$.

STEP 2. Because node 4 is not the sink node, go to STEP 1.

STEP 1. Because $\text{Lab}(V_{v,\text{Pos}(V_v)}) = \text{Lab}(V_{4,1}) = \text{Lab}(2)=0$, let $\text{Pos}(V_v) = \text{Pos}(V_v)+1=2$ and go to STEP 4.

STEP 4. Because $\text{Pos}(V_v) =2 < |V_v| =3$, go to STEP 1.

STEP 1. Because $\text{Lab}(V_{v,\text{Pos}(V_v)}) = \text{Lab}(V_{4,2}) = \text{Lab}(5)=0$, let $\text{Lab}(5)=1$, $i = i+1=5$, $v = v_i = v_5 =5$ (see Fig. 6), and $\text{Pos}(V_v) =1$.

STEP 2. Because node 5 is not the sink node, go to STEP 1.

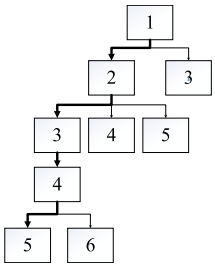

FIGURE 6. The DFS-tree with offspring node 5 of Figure 1.

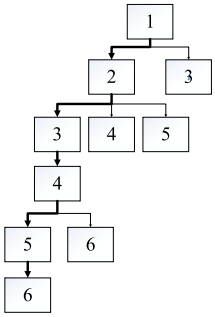

FIGURE 7. The DFS-tree with offspring node 6 of Figure 1.

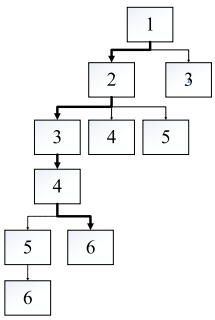

FIGURE 8. The DFS-tree with offspring node 6 of Figure 1.

STEP 1. Because $\text{Lab}(V_{v,\text{Pos}(V_v)}) = \text{Lab}(V_{5,1}) = \text{Lab}(4) = 0$, let $\text{Pos}(V_v) = \text{Pos}(V_v) + 1 = 2$ and go to STEP 4.

STEP 4. Because $\text{Pos}(V_v) = |V_v| = 2$, go to STEP 1.

STEP 1. Because $\text{Lab}(V_{v,\text{Pos}(V_v)}) = \text{Lab}(V_{5,2}) = \text{Lab}(6) = 0$, let $\text{Lab}(6) = 1$, $i = i+1 = 6$, $v = v_i = v_6 = 6$ (see Fig. 7), and $\text{Pos}(V_v) = 1$.

STEP 2. Because node 6 is the sink node, we have a new MPN—say, X_1—$\{v_1, v_2, \ldots, v_6\} = \{1, 2, 3, 4, 5, 6\}$; i.e., an MP: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$.

STEP 3. Let $\text{Lab}(v_i) = \text{Lab}(6) = 0$, $i = i-1 = 5$, and $v = v_i = 5$.

STEP 4. Because $\text{Pos}(V_v) = |V_v| = 2$, go to STEP 3.

STEP 3. Let $\text{Lab}(v_i) = \text{Lab}(5) = 0$, $i = i-1 = 4$, and $v = v_i = 4$.

STEP 4. Because $\text{Pos}(V_v) = 2 < |V_v| = 3$, let $\text{Pos}(V_v) = \text{Pos}(V_v) + 1 = 3$ and go to STEP 1.

STEP 1. Because $\text{Lab}(V_{v,\text{Pos}(V_v)}) = \text{Lab}(V_{4,3}) = \text{Lab}(6) = 0$, let $\text{Lab}(6) = 1$, $i = i+1 = 5$, $v = v_i = v_5 = 6$ (see Fig. 8), and $\text{Pos}(V_v) = 1$.

TABLE 1. Final results of the step-by-step example.

k	MPN	the corresponding MP
1	{1, 2, 3, 4, 5, 6}	$\{e_{12}, e_{23}, e_{34}, e_{45}, e_{56}\}$
2	{1, 2, 3, 4, 6}	$\{e_{12}, e_{23}, e_{34}, e_{46}\}$
3	{1, 2, 4, 5, 6}	$\{e_{12}, e_{24}, e_{45}, e_{56}\}$
4	{1, 2, 4, 6}	$\{e_{12}, e_{24}, e_{46}\}$
5	{1, 2, 5, 4, 6}	$\{e_{12}, e_{25}, e_{54}, e_{46}\}$
6	{1, 2, 5, 6}	$\{e_{12}, e_{25}, e_{56}\}$
7	{1, 3, 2, 4, 5, 6}	$\{e_{13}, e_{32}, e_{24}, e_{45}, e_{56}\}$
8	{1, 3, 2, 4, 6}	$\{e_{13}, e_{32}, e_{24}, e_{46}\}$
9	{1, 3, 2, 5, 4, 6}	$\{e_{13}, e_{32}, e_{25}, e_{54}, e_{46}\}$
10	{1, 3, 2, 5, 6}	$\{e_{13}, e_{32}, e_{25}, e_{56}\}$
11	{1, 3, 4, 2, 5, 6}	$\{e_{13}, e_{34}, e_{42}, e_{25}, e_{56}\}$
12	{1, 3, 4, 5, 6}	$\{e_{13}, e_{34}, e_{45}, e_{56}\}$
13	{1, 3, 4, 6}	$\{e_{13}, e_{34}, e_{46}\}$

STEP 2. Because node 6 is the sink node, we have a new MPN—say, X_2—$\{v_1, v_2, \ldots, v_5\} = \{1, 2, 3, 4, 6\}$; i.e., an MP: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6$.

:

:

The final result for finding the complete MPN set without duplicates using the proposed algorithm is listed in Table 1:

VI. THE COMPUTATIONAL EXPERIMENTS

In this study, two experiments are conducted to test the performance of the proposed algorithm: Ex1 and Ex2. Both experiments are coded in the C programming language and implemented on an Intel Core i7-5960X CPU 3.00 GHz with 16 GB of RAM and 64-bit Windows 10.

A. Ex1: TEST ON 20 BENCHMARK PROBLEMS

In Ex1, two algorithms are tested on 20 conventional benchmark networks, as shown in Figure 9, after removing the directions of the edges: the proposed algorithm and the conventional edge-based algorithms proposed in [7], [43]–[48].

Table 2 lists the runtime (in CPU seconds) for each benchmark problem solved using both algorithms. T_{node} and T_{edge} are the related times for the proposed node-based MP algorithm and the edge-based MP algorithm, respectively. The test for the related benchmark problem is terminated if its runtime is more than 1 h.

It takes only $O(|p|)$ to find any MP p. Hence, the run time is very close to zero if the number of MPs or edges are very small. Therefore, as seen in Table 2, $T_{\text{node}} = T_{\text{edge}} = 0$ for these benchmark problems with a number of edges (i.e., $|E|$) less than or equal to 14—i.e., benchmark problems 1–4. However, T_{edge} increases exponentially with increasing $|E|$ because it is based on edge subsets and the NP-hard characteristic of the MP problem. In contrast, T_{node} is still zero for the remainder of any benchmark problem. Thus, the proposed node-based MP algorithm is more efficient than the conventional edge-based MP algorithm.

The foregoing observation is the same as the time complexity analyzed in Theorem 4 of Section 3. Thus, the proposed node-based MP algorithm outperforms the conventional edge-based MP algorithm from both theoretical and practical aspects.

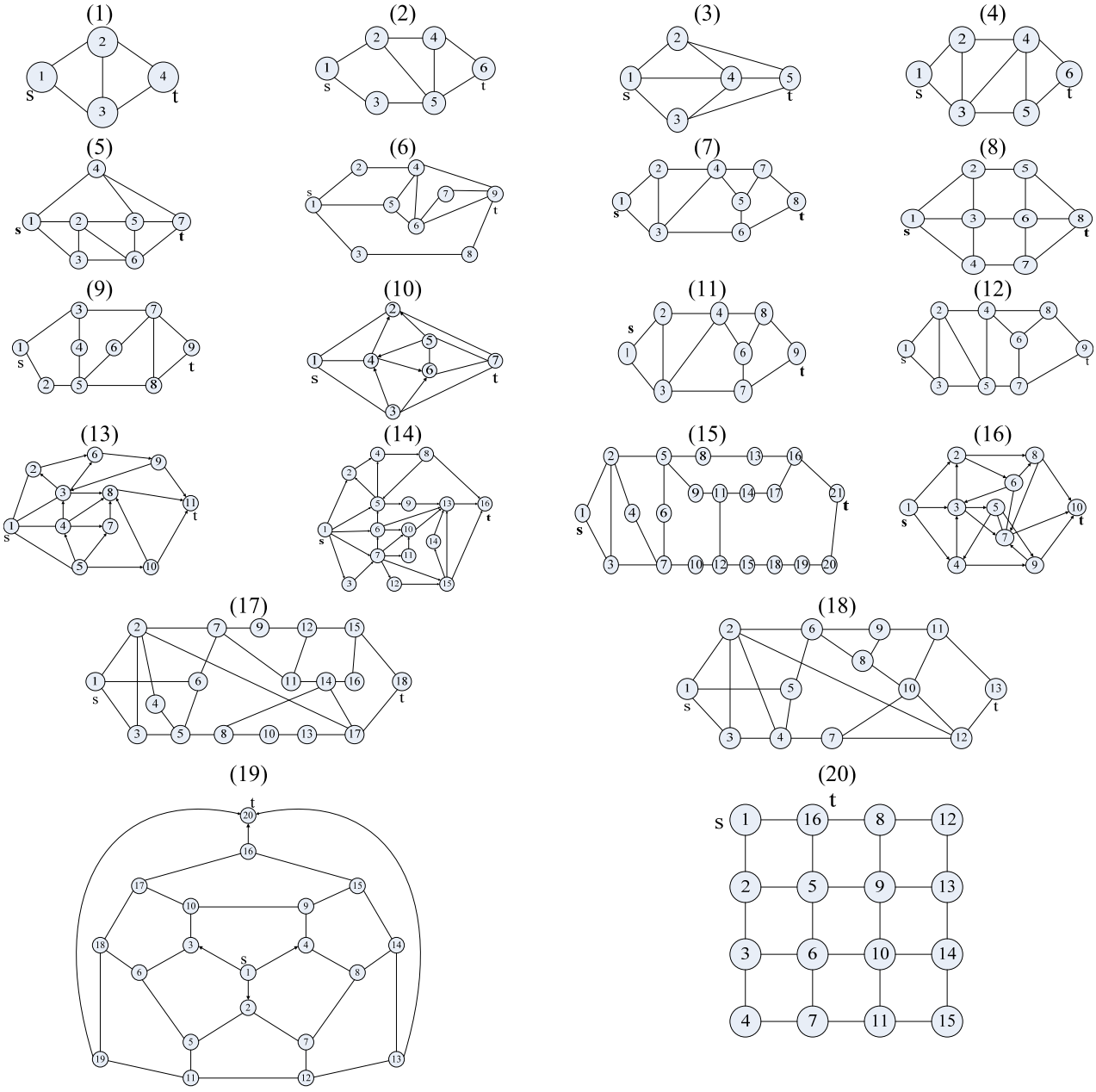

FIGURE 9. Benchmark networks.

B. Ex2: TEST ON COMPLETE NETWORKS

Each complete network has the maximal number of edges compared to the network with the same number of nodes. For example, Fig. 10 shows a complete graph with 12 nodes.

In Ex2, the experiment is extended to larger complete graphs to verify the maximal node number with which the proposed algorithm is still able to find all MPs. The results are listed in Tables 3 and 4.

In Table 3, $|E| = |V| \cdot (|V| - 1)/2$ from the definition of complete networks. In addition, from Table 3, the runtime increases exponentially with the number of nodes, which concurs with the NP-hard characteristic for the network reliability problems. The maximal node number in this test is $|V| = 16$ because the runtime for 17-node completion is more than 24 h and terminated.

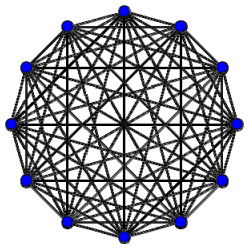

FIGURE 10. A 12-node complete graph (https://en.wikipedia.org/wiki/Complete_graph).

TABLE 2. The CPU seconds spent and the number of obtained MPs for the two algorithms.

ID	$\|V\|$	$\|E\|$	$\|P\|$	T_{node}	T_{edge}
1	4	6	4	0.0000000000	0.0000000000
2	6	12	7	0.0000000000	0.0000000000
3	5	10	9	0.0000000000	0.0000000000
4	6	14	13	0.0000000000	0.0000000000
5	7	18	25	0.0000000000	0.0309999995
6	9	19	12	0.0000000000	0.0460000001
7	8	20	24	0.0000000000	0.1089999974
8	8	20	29	0.0000000000	0.1089999974
9	9	20	13	0.0000000000	0.1129999980
10	7	21	52	0.0000000000	0.1860000044
11	9	24	40	0.0000000000	2.1089999676
12	9	24	44	0.0000000000	2.1089999676
13	11	35	151	0.0000000000	3600.0061035156
14	16	48	257	0.0000000000	3600.0129394531
15	21	48	44	0.0000000000	3600.0129394531
16	10	36	331	0.0000000000	3600.0129394531
17	18	49	269	0.0000000000	3600.0129394531
18	13	39	269	0.0000000000	3600.0129394531
19	20	54	780	0.0000000000	3600.0129394531
20	16	43	44	0.0000000000	3600.0129394531

TABLE 3. The obtained runtimes and the number of MPs for complete networks.

k=$\|V\|$	$\|E_k\|$	$\|P_k\|$	$T_{\text{node},k}$
5	10	16	0.000000000000000
6	15	65	0.000000000000000
7	21	326	0.000000000000000
8	28	1957	0.000000000000000
9	36	13700	0.000000000000000
10	45	109601	0.004999999888241
11	55	986410	0.050000000745058
12	66	9864101	0.529999971389771
13	78	108505112	6.136000156402588
14	91	1302061345	76.871002197265625
15	105	16926797486	1037.279052734375000
16	120	236975164805	15150.948242187500000
17	136	3554627472076	236083.937500000000000

TABLE 4. The obtained ratios for complete networks.

k	$k/(k+1)$	$\|E_k\|/\|E_{k+1}\|$	$\|P_k\|/\|P_{k+1}\|$	$\|T_{\text{node},k}\|/\|T_{\text{node},k+1}\|$
5				
6	1.200	1.5000	4.062500000000	
7	1.167	1.4000	5.015384615385	
8	1.143	1.3333	6.003067484663	
9	1.125	1.2857	7.000510986203	
10	1.111	1.2500	8.000072992701	
11	1.100	1.2222	9.000009124004	10.000000372530
12	1.091	1.2000	10.000001013777	10.599999269843
13	1.083	1.1818	11.000000101378	11.577359410629
14	1.077	1.1667	12.000000009216	12.527868356890
15	1.071	1.1538	13.000000000768	13.493762577370
16	1.067	1.1429	14.000000000059	14.606434210975
17	1.063	1.1333	15.000000000004	15.582122895954

In Table 4, the ratio of values of two consecutive items—i.e., $k/(k+1)$, $|E_k|/|E_{k+1}|$, $|P_k|/|P_{k+1}|$, and $|T_{\text{node},k}|/|T_{\text{node},k+1}|$—are listed to observe the trend and relationship among these items. Note that these ratios have empty values because their denominators are zero. From Table 4, the ratio of $|P_k|/|P_{k+1}|$ and the value $k-2$ tend to coincide; e.g., $|P_{13}|/|P_{14}|$ is almost equal to 12, and there is a similar result for the ratio of $|T_k|/|T_{k+1}|$ and the value $k-2$. These two significant observations are able to predict the maximum number of MPs and the runtime to search for all MPs for complete networks and even for general networks.

In addition, the ratios $|P_k|/|P_{k+1}|$ and $|T_{\text{node},k}|/|T_{\text{node},k+1}|$ are almost equal; e.g., both $|P_{13}|/|P_{14}|$ and $|T_{\text{node},13}|/|T_{\text{node},14}|$ are near 12.

Hence, the runtime of the proposed node-based MP algorithm linearly increases with the increasing number of MPs. Thus, the proposed algorithm is very efficient for the network reliability NP-hard problem.

VII. CONCLUSION

This work proposes a simple and efficient novel concept for finding all MPs, which is an NP-hard problem. The novel concept can be implemented with time complexity $O(ne^n)$ by finding all MPs from ordered node subsets. It is a significant improvement over the previous $O(n2^m)$ time limits [29-34], where $O(m) = O(n^2)$ [48]. In addition, a node-based MP algorithm is proposed to efficiently implement the node-based concept without testing each ordered node subset to find all MPs. From an extensive experimental study on 20 benchmark problems in Ex1, the proposed node-based MP algorithm is clearly superior to and more efficient than the conventional edge-based MP algorithm. In Ex2, the proposed algorithm is able to search for all of the MPs in these complete networks up to 16 nodes, and the runtime to find MPs increases linearly with the number of MPs. Hence, the proposed novel concept is very useful for improving the efficiency of searching for all MPs theoretically and practically.

REFERENCES

[1] S. Rai and K. K. Aggarwal, "On complementation of pathsets and cutsets," *IEEE Trans. Rel.*, vol. R-29, no. 2, pp. 139–140, Jun. 1980.
[2] Y. Shen, "A new simple algorithm for enumerating all minimal paths and cuts of a graph," *Microelectron. Rel.*, vol. 35, no. 6, pp. 973–976, 1995.
[3] S. Song, D. W. Coit, Q. Feng, and H. Peng, "Reliability analysis for multi-component systems subject to multiple dependent competing failure processes," *IEEE Trans. Rel.*, vol. 63, no. 1, pp. 331–345, Mar. 2014.
[4] T. Aven, "Some considerations on reliability theory and its applications," *Rel. Eng. Syst. Saf.*, vol. 21, no. 3, pp. 215–223, 1988.
[5] C. J. Colbourn, *The Combinatorics of Network Reliability*. New York, NY, USA: Oxford Univ. Press, 1987.
[6] G. Levitin, *The Universal Generating Function in Reliability Analysis and Optimization*. London, U.K.: Springer-Verlag, 2005.
[7] D. R. Shier, *Network Reliability and Algebraic Structures*. New York, NY, USA: Clarendon Press, 1991.
[8] H. Pham, "Special issue on critical reliability challenges and practices [guest editorial]," *IEEE Trans. Syst., Man, Cybern.-A, Syst. Hum.*, vol. 37, no. 2, pp. 141–142, Mar. 2007.
[9] S.-C. Wei and W.-C. Yeh, "Resource allocation decision model for dependable and cost-effective grid applications based on Grid Bank," *Future Gener. Comput. Syst.*, vol. 77, pp. 12–28, Dec. 2017.
[10] W.-J. Ke and S.-D. Wang, "Reliability evaluation for distributed computing networks with imperfect nodes," *IEEE Trans. Rel.*, vol. 46, no. 3, pp. 342–349, Sep. 1997.
[11] M. Wang, W.-C. Yeh, T.-C. Chu, X. Zhang, C.-L. Huang, and J. Yang, "Solving multi-objective fuzzy optimization in wireless smart sensor networks under uncertainty using a hybrid of IFR and SSO algorithm," *Energies*, vol. 11, no. 9, p. 2385, 2018.
[12] C. Lin, L. Cui, D. W. Coit, and M. Lv, "Performance analysis for a wireless sensor network of star topology with random nodes deployment," *Wireless Pers. Commun.*, vol. 97, no. 3, pp. 3993–4013, 2017.
[13] C. L. Huang, S.-Y. Huang, W. C. Yeh, and J. Wang, "Fuzzy system and time window applied to traffic service network problems under a multi-demand random network," *Electron*, vol. 8 no. 5, p. 539, 2019, doi: 10.3390/electronics8050539.

[14] R. Moghaddass, M. J. Zuo, and J. Qu, "Reliability and availability analysis of a repairablek-out-of-*n*: *G*System with*R*Repairmen subject to shut-Off rules," *IEEE Trans. Rel.*, vol. 60, no. 3, pp. 658–666, Sep. 2011.
[15] J. Wang, W.-C. Yeh, N. N. Xiong, J. Wang, X. He, and C. L. Huang, "Building an improved Internet of things smart sensor network based on a three-phase methodology," *IEEE Access*, vol. 7, pp. 141728–141737, 2019, doi: 10.1109/ACCESS.2019.2925044.
[16] W.-C. Yeh and J.-S. Lin, "New parallel swarm algorithm for smart sensor systems redundancy allocation problems in the Internet of Things," *J. Supercomput.*, vol. 74, no. 9, pp. 4358–4384, 2018.
[17] C.-L. Huang, "A particle-based simplified swarm optimization algorithm for reliability redundancy allocation problems," *Rel. Eng. Syst. Saf.*, vol. 142, pp. 221–230, Oct. 2015.
[18] W.-C. Yeh, "A squeezed artificial neural network for the symbolic network reliability functions of binary-state networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2822–2825, Nov. 2017.
[19] C.-M. Lai and W.-C. Yeh, "Two-stage simplified swarm optimization for the redundancy allocation problem in a multi-state bridge system," *Rel. Eng. Syst. Saf.*, vol. 156, pp. 148–158, Dec. 2016.
[20] L. Xing, "Reliability evaluation of phased-mission systems with imperfect fault coverage and common-cause failures," *IEEE Trans. Rel.*, vol. 56, no. 1, pp. 58–68, Mar. 2007.
[21] W.-C. Yeh, "Solving cold-standby reliability redundancy allocation problems using a new swarm intelligence algorithm," *Appl. Soft Comput.*, vol. 83, Oct. 2019, Art. no. 105582.
[22] W. Zhu, W. C. Yeh, L. Cao, Z. Zhu, D. Chen, J. Chen, A. Li, and Y. Lin, "Faster Evolutionary Convolutional Neural Networks Based on iSSO for Lesion Recognition in Medical Images," *Basic Clin. Pharmacol. Toxicol.*, vol. 124, p. 329, Apr. 2019.
[23] W.-C. Yeh and M. J. Zuo, "A new subtraction-based algorithm for the d-MPs for all d problem," *IEEE Trans. Rel.*, vol. 68, no. 3, pp. 999–1008, Sep. 2019.
[24] Z. Hao, W.-C. Yeh, and C.-F. Hu, "A novel multistate minimal cut vectors problem and its algorithm," *IEEE Trans. Rel.*, vol. 68, no. 1, pp. 291–301, Mar. 2019.
[25] W.-C. Yeh and T.-C. Chu, "A novel multi-distribution multi-state flow network and its reliability optimization problem," *Rel. Eng. Syst. Saf.*, vol. 176, pp. 209–217, Aug. 2018.
[26] W.-C. Yeh, "A novel boundary swarm optimization method for reliability redundancy allocation problems," *Rel. Eng. Syst. Saf.*, to be published, doi: 10.1016/j.ress.2018.02.002.
[27] W.-C. Yeh, "Fast Algorithm for Searching *d* -MPs for all Possible d," *IEEE Trans. Rel.*, vol. 67, no. 1, pp. 308–315, Mar. 2018.
[28] W.-C. Yeh, "Methodology for the reliability evaluation of the novel learning-effect multi-state flow network," *IISE Trans.*, vol. 49, no. 11, pp. 1078–1085, 2017.
[29] W.-C. Yeh, "A new exact solution algorithm for a novel generalized redundancy allocation problem," *Inf. Sci.*, vol. 408, pp. 182–197, Oct. 2017.
[30] W.-C. Yeh, "Evaluation of the one-to-all-target-subsets reliability of a novel deterioration-effect acyclic multi-state information network," *Rel. Eng. Syst. Saf.*, vol. 166, pp. 132–137, Oct. 2017.
[31] W.-C. Yeh, "A fast algorithm for quickest path reliability evaluations in multi-state flow networks," *IEEE Trans. Rel.*, vol. 64, no. 4, pp. 1175–1184, Dec. 2015.
[32] W.-C. Yeh, C. Bae, and C.-L. Huang, "A new cut-based algorithm for the multi-state flow network reliability problem," *Rel. Eng. Syst. Saf.*, vol. 136, pp. 1–7, Apr. 2015.
[33] W.-C. Yeh, "A novel node-based sequential implicit enumeration method for finding all d-MPs in a multistate flow network," *Inf. Sci.*, vol. 297, pp. 283–292, Mar. 2015.
[34] W.-C. Yeh, "Orthogonal simplified swarm optimization for the series–parallel redundancy allocation problem with a mix of components," *Knowl.-Based Syst.*, vol. 64, pp. 1–12, Jul. 2014.
[35] Z. Hao, W.-C. Yeh, J. Wang, G.-G. Wang, and B. Sun, "A quick inclusion-exclusion technique," *Inf. Sci.*, 486, pp. 20–30, Jun. 2019.
[36] W.-C. Yeh, "An improved sum-of-disjoint-products technique for symbolic multi-state flow network reliability," *IEEE Trans. Rel.*, vol. 64, no. 4, pp. 1185–1193, Dec. 2015.
[37] H. George-Williams and E. Patelli, "A hybrid load flow and event driven simulation approach to multi-state system reliability evaluation," *Rel. Eng. Syst. Saf.*, vol. 152, pp. 351–367, Aug. 2016.
[38] J. E. Ramirez-Marquez and D. W. Coit, "A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability," *Rel. Eng. Syst. Saf.*, vol. 87, no. 2, pp. 253–264, 2005.
[39] J. E. Ramirez-Marquez, "Assessment of the transition-rates importance of Markovian systems at steady state using the unscented transformation," *Rel. Eng. Syst. Saf.*, vol. 142, pp. 212–220, Oct. 2015.
[40] T. Zhang and B. Guo, "Capacitated stochastic coloured Petri net-based approach for computing two-terminal reliability of multi-state network," *J. Syst. Eng. Electron.*, vol. 23, no. 2, pp. 304–313, Apr. 2012.
[41] Y.-F. Niu and F.-M. Shao, "A practical bounding algorithm for computing two-terminal reliability based on decomposition technique," *Comput. Math. Appl.*, vol. 61, no. 8, pp. 2241–2246, 2011.
[42] J. E. Ramírez-Márquez and W. Jiang, "Confidence bounds for the reliability of binary capacitated two-terminal networks," *Rel. Eng. Syst. Saf.*, vol. 91, no. 8, pp. 905–914, 2006.
[43] W.-C. Yeh, "New method in searching for all minimal paths for the directed acyclic network reliability problem," *IEEE Trans. Rel.*, vol. 65, no. 3, pp. 1263–1270, Sep. 2016.
[44] S.-G. Chen and Y.-K. Lin, "Search for all minimal paths in a general large flow network," *IEEE Trans. Rel.*, vol. 61, no. 4, pp. 949–956, Dec. 2012.
[45] Y. F. Niu, Z. Y. Gao, and H. J. Sun, "An improved algorithm for solving all d-MPs in multi-state networks," *J. Syst. Sci. Syst. Eng.*, vol. 26, no. 6, pp. 711–731, 2017.
[46] W.-C. Yeh, "A simple heuristic algorithm for generating all minimal paths," *IEEE Trans. Rel.*, vol. 56, no. 3, pp. 488–494, Sep. 2007.
[47] W.-C. Yeh, "A simple universal generating function method to search for all minimal paths in networks," *IEEE Trans. Syst., Man, Cybern.-A, Syst. Hum.*, vol. 39, no. 6, pp. 1247–1254, Nov. 2009.
[48] W.-C. Yeh, "A simple algorithm to search for all MCs in networks," *Eur. J. Oper. Res.*, vol. 174, no. 3, pp. 1694–1705, 2006.
[49] Y. Niu, Z. Gao, and W. H. K. Lam, "A new efficient algorithm for finding all d-minimal cuts in multi-state networks," *Rel. Eng. Syst. Saf.*, vol. 166, pp. 151–163, Oct. 2017.

WENBO ZHU received the M.S. degree from the Department of Automation, Harbin Engineering University, Harbin, China, and the Ph.D. degree from the Department of Automation Science and Engineering, South China University of Technology, Guangzhou, China. He is currently a Lecturer with the Department of Automation, Foshan University, Foshan, China. He has proposed some original articles in journals, such as *Signal Processing*, the *Journal of Medical Imaging and Health Informatics*, and *Computers in Biology and Medicine*. His current research interest includes artificial intelligence algorithms, including pattern recognition, machine learning, and evolutionary computation.

WEI-CHANG YEH received the M.S. and Ph.D. degrees from the Department of Industrial Engineering, The University of Texas at Arlington. He is currently a Distinguished Professor with the Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Taiwan. He has published more than 250 research articles in highly ranked journals and conference papers and has been received the Outstanding Research Award twice, the Distinguished Scholars Research Project once, and the Overseas Research Fellowship twice from the Ministry of Science and Technology, Taiwan. His current research interest includes algorithms, including exact solution methods and soft computing. He proposed a novel soft computing algorithm called the simplified swarm optimization (SSO) and demonstrated the simplicity, effectiveness, and efficiency of his SSO for solving NP-hard problems. He has been granted 60 patents and received an International Fellowship, the Guoguang Invention Medal, and the titles of Outstanding Inventor of Taiwan and Doctor of Erudition, by the Chinese Innovation and Invention Society. He has been invited to serve as an Associate Editor for the two top reliability related journals, namely, the IEEE TRANSACTIONS ON RELIABILITY AND RELIABILITY ENGINEERING & SYSTEM SAFETY.

NEAL NAIXUE XIONG received the Ph.D. degree in software engineering from Wuhan University and the Ph.D. degree in dependable networks from the Japan Advanced Institute of Science and Technology. He was with the Wentworth Technology Institution, Georgia State University, for many years. He is currently an Associate Professor with the Department of Mathematics and Computer Science, Northeastern State University. His current research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory.

BIN SUN received the B.S. degree in information and computer science and the Ph.D. degree in control science and engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005 and 2010, respectively. She is currently an Associate Professor with the School of Aeronautics and Astronautics, University of Electronic Science and Technology of China. She is a member of the Aircraft Swarm Intelligent Sensing and Cooperative Control Key Laboratory of Sichuan. Her current research interests include information fusion, tracking, and guidance.