Hindawi
Complexity
Volume 2020, Article ID 8060794, 10 pages
https://doi.org/10.1155/2020/8060794

Research Article

A Minimal Path-Based Method for Computing Multistate Network Reliability

Xiu-Zhen Xu, Yi-Feng Niu, and Can He

School of Economics and Management, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Correspondence should be addressed to Xiu-Zhen Xu; jluxxz@qq.com

Received 4 May 2020; Revised 13 September 2020; Accepted 30 September 2020; Published 22 October 2020

Academic Editor: Ning Cai

Copyright © 2020 Xiu-Zhen Xu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Most of modern technological networks that can perform their tasks with various distinctive levels of efficiency are multistate networks, and reliability is a fundamental attribute for their safe operation and optimal improvement. For a multistate network, the two-terminal reliability at demand level d, defined as the probability that the network capacity is greater than or equal to a demand of d units, can be calculated in terms of multistate minimal paths, called d-minimal paths (d-MPs) for short. This paper presents an efficient algorithm to find all d-MPs for the multistate two-terminal reliability problem. To advance the solution efficiency of d-MPs, an improved model is developed by redefining capacity constraints of network components and minimal paths (MPs). Furthermore, an effective technique is proposed to remove duplicate d-MPs that are generated multiple times during solution. A simple example is provided to demonstrate the proposed algorithm step by step. In addition, through computational experiments conducted on benchmark networks, it is found that the proposed algorithm is more efficient.

1. Introduction

The economic development and social well-being of modern society are increasingly dependent on the quality of service (QoS) of a variety of technological networks, such as computer and communication networks [1, 2], manufacturing networks [3], logistics/supply chain networks [4–7], and power transmission networks [8], and a major challenge which we have to face is ensuring their safety and normal operation against potential malfunctions. Reliability is a fundamental attribute for the safe operation of modern technological networks [9], and demands for the quantification of network reliability are steadily growing in the networked society [10].

A network that can have a number of distinctive levels of performance is called multistate network [11, 12]. Most of modern technological networks are often deemed as multistate networks for reliability analysis. For instance, a computer network can be modeled as a multistate network with a set of edges (links) and a set of nodes, in which an edge represents a transmission line and a node denotes a computer center involving several routers or switches. In practice, a transmission line is composed of several physical lines, such as fiber cables, twisted pairs, or coaxial cables, and all of the physical lines may provide a specific capacity or fail, that is, every transmission line has multiple capacities (states) with a probability distribution, which results in a number of different levels of transmission capacity for the whole computer network. Such a computer network is thus treated as a typical multistate network. Multistate network with components having many different states (different levels of performance) is an extension of the traditional binary-state network with components having only two possible states: either completely failed or perfectly functioning [13–16]. Given a multistate network, the two-terminal reliability at demand level d, denoted as R_d, is defined as the probability that the network capacity is greater than or equal to a demand of d units. A variety of algorithms have been presented to calculate R_d, and these algorithms can be broadly categorized into direct methods and indirect methods [17]. One important type of indirect method for the exact evaluation of R_d is using d-minimal paths (d-MPs). A d-MP, say x, is a minimal network state vector with the max-flow of the network under x being equal to d. Provided

that all d-MPs are found, the well-known Inclusion-Exclusion method [14], the Sum of Disjoint Products method [18–20], or the State Space Decomposition method [21] are available to exactly compute the two-terminal reliability at demand level d. Thus, solving all d-MPs is a critical issue to the two-terminal reliability at demand level d.

There are two important mathematical models with respect to the d-MP problem. The first mathematical model for solving d-MPs is originally proposed by Lin et al. [22]. This model requires all minimal paths (MPs) and contains three constraints that are established by the network structure and the flow-conservation law. An MP is a subset of edges, such that if any edge is removed from it, the remaining is no longer a path. The second mathematical model for solving d-MPs is proposed by Yeh [23]. This model without requiring MPs information is a component-based formulation with three constraints. Note that the model by Yeh [23] is derived from the well-known max-flow mathematical programming model; thus, it is more applicable to directed networks. Based on the model of Yeh, Xu et al. [24] combined the max-flow algorithm and the implicit enumeration method to seek d-MPs. Niu et al. [25] recently improved the model by Yeh [23] and pointed out that if the model is applied to an undirected network, each undirected edge in the network should be treated as two directed edges with the opposite direction. In such a case, however, the number of variables in the model is dramatically increased, which could significantly add to the difficulty of the d-MP problem.

The model by Lin et al. [22] applicable to both directed networks and undirected networks has been widely used by most of the existing algorithms [26–30]. For example, Lin [26] proposed a simple method to solve d-MPs of a network with unreliable nodes; Yeh [27] proposed a cycle-checking method to verify whether a feasible solution to the model is a d-MP; Chen and Lin [28] suggested the fast enumeration method to solve all of the feasible solutions to the model; Forghani-Elahabad and Bonani [29] proposed an improved enumeration method to search for d-MPs. Lin and Chen [30] recently proposed a new max-flow evaluation method to find d-MPs.

In addition to the methods based on the above two models, there are also several other methods for solving d-MPs. For example, Satitsatian and Kapur [31] proposed an approximate algorithm to find a subset of d-MPs using the minimal improvement paths algorithm; Ramirez-Marquez et al. [32] proposed an algorithm to find potential d-MPs using the information sharing approach that a selected number of edges share information among each other. Different from the abovementioned methods that find d-MPs for a fixed demand level d, Bai et al. [33] considered the problem of finding all d-MPs for all possible demand levels and proposed a recursive method to solve it. By overcoming the obstacles of the method by Bai et al. [33], Yeh [34] recently proposed a new addition-based algorithm with better time complexity.

As stated earlier, the model by Lin et al. [22] has been extensively applied to the d-MP problem. And yet, we note that the capacity constraints in this model are too relaxed, such that a large number of enumerations need to be implemented in order to find all d-MPs, which to an extent influences the computational efficiency. Recently, Niu et al. [25] improved Yeh's model [23] by imposing stronger capacity constraints on network components. Because the capacity constraints of network components in both models are identical, the ideas in [25] can also be used to improve the model by Lin et al. [22]. In addition, the major cost for solving d-MPs from the model by Lin et al. [22] lies in identifying duplicate d-MPs, and an effective and efficient method for removing duplicates is always desirable. This paper is devoted to developing a new efficient algorithm for finding all d-MPs. To advance the solution efficiency of d-MPs, an improved model is constructed by redefining capacity constraints of network components and MPs. The constructed model is an improvement to the model by Lin et al. [22]. Furthermore, a technique with better efficiency is proposed to remove duplicate d-MPs. The time complexity of the proposed algorithm is analyzed, and a simple example is provided to illustrate the proposed algorithm. Finally, through computational experiments, it is found that the proposed algorithm is more efficient in finding all d-MPs.

The rest of this paper is organized as follows. Section 2 introduces the multistate network model, the fundamental results for solving d-MPs, and the Inclusion-Exclusion method. In Section 3, an improved model for finding d-MPs is constructed, and an efficient technique is developed to remove duplicate d-MPs. Also, an algorithm for solving d-MPs without duplicates is proposed, together with a discussion on its time complexity. In Section 4, an illustrative example is provided to demonstrate the proposed algorithm. Computational experiments on two benchmark networks are performed in Section 5 to investigate the efficiency of the proposed algorithm through comparisons with the existing methods. Section 6 presents the concluding remarks.

2. Preliminaries

2.1. Multistate Network. Consider a multistate network $G(V, E, W)$ with the unique source node s and the unique sink node t, where $V = \{s, 1, 2, \ldots, n, t\}$ is the set of nodes, $E = \{e_1, e_2, \ldots, e_m\}$ is the set of edges, and $W = (W_1, W_2, \ldots, W_m)$ is the largest state vector with W_i being the largest capacity of e_i for $1 \le i \le m$. A network state vector $x = (x_1, x_2, \ldots, x_m)$ indicates the current capacity of each edge in the network, where the state of e_i is defined by x_i taking integer values from 0 to W_i. The max-flow of the network under x (or the network capacity under x) is denoted by $M(x)$, and $M(x)$ is always called the structure function of a multistate network [25, 31, 35]. By D, the max-flow of the network under the largest state vector W is denoted, i.e., $D = M(W)$. Then, $M(x) \le D$ holds for any state vector x.

The multistate network discussed herein satisfies the following assumptions [28, 30]. (1) Each node is perfectly reliable, which means no capacity constraint is imposed on nodes. (2) The state/capacity of each edge e_i $(1 \le i \le m)$ is a nonnegative integer-valued random variable which takes values from 0 to W_i according to a given distribution. (3) The

states/capacities of different edges are statistically independent. (4) All flows in the network obey the conservation law, i.e., total flows into and from a node (not source and sink nodes) are all equal.

2.2. Calculating R_d in Terms of D-MPs. As mentioned before, once all d-MPs are found, the well-known Inclusion-Exclusion method [14], the Sum of Disjoint Products method [18–20], or the State Space Decomposition method [21] are available to calculate R_d. While the Inclusion-Exclusion method is not as efficient as the Sum of Disjoint Products method [18–20] or the State Space Decomposition method [21], it is simple and easy to understand. Therefore, we briefly introduce how it performs in calculating R_d. Assume $y^1, y^2, \ldots, y^\sigma$ are all d-MPs, and let $B_1 = \{x|x \geq y^1\}, B_2 = \{x|x \geq y^2\}, \ldots, B_\sigma = \{x|x \geq y^\sigma\}$, where $x = (x_1, x_2, \ldots, x_m)$, $y^i = (y^i_1, y^i_2, \ldots, y^i_m)$, and $x \geq y^i$ means that $x_j \geq y^i_j$ for $j = 1, 2, \ldots, m$; then, R_d can be calculated via the Inclusion-Exclusion method as follows:

$$R_d = \Pr(B_1 \cup B_2 \cup \cdots \cup B_\sigma) = \sum_{i=1}^{\sigma} \Pr(B_i) - \sum_{j=2}^{\sigma} \sum_{i=1}^{j-1} \Pr(B_i \cap B_j) + \cdots + (-1)^{\sigma-1} \Pr(B_1 \cap B_2 \cap \cdots \cap B_\sigma), \tag{1}$$

where

$$\begin{aligned} \Pr\{B_i\} &= \Pr\{x|x \geq y^i\} = \prod_{k=1}^{m} \Pr\{x_k \geq y^i_k\}, \Pr\{B_i \cap B_j\} \\ &= \Pr\{x|x \geq \max\{y^i, y^j\}\} \\ &= \prod_{k=1}^{m} \Pr\{x_k \geq \max\{y^i_k, y^j_k\}\}, \ldots, \Pr(B_1 \cap B_2 \cap \cdots \cap B_\sigma) \\ &= \Pr\{x|x \geq \max\{y^1, y^2, \ldots, y^\sigma\}\} \\ &= \prod_{k=1}^{m} \Pr\{x_k \geq \max\{y^1_k, y^2_k, \ldots, y^\sigma_k\}\}. \end{aligned} \tag{2}$$

In addition, it should be pointed out that when the number of d-MPs is huge, the Sum of Disjoint Products method [18–20] or the State Space Decomposition method [21] is applicable to the calculation of R_d. Readers are encouraged to refer to [18–21] for the details of the two methods.

2.3. The Fundamental Results for Solving d-MPs. A state vector x is a d-MP if and only if (1) $M(x) = d$ and (2) $M(x - 0(e_i)) < d$ for each $x_i > 0$, where $0(e_i) = (0, \ldots, 0, 1, 0, \ldots, 0)$, i.e., capacity is 1 for e_i and 0 for other edges, that is, two conditions must be fulfilled for a d-MP x. The first condition suggests that the network capacity under x is equal to d, and the second reveals that capacity degradation of any edge with nonzero capacity would lead to a smaller network capacity (below d). The above definition implicitly demonstrates that a d-MP is also the minimal state vector satisfying the demand level d.

The well-known model proposed by Lin et al. [22] is one of the two fundamental models with respect to d-MPs. Suppose that there are totally p MPs, $P_1, P_2, \ldots, P_p$, from the source node s to the sink node t in the network, and the flow travelling through P_j $(1 \leq j \leq p)$ is denoted by f_j $(1 \leq j \leq p)$. A flow vector $f = (f_1, f_2, \ldots, f_p)$ consists of flows travelling through all MPs. The following model by Lin et al. [22] is the foundation of most of the existing algorithms.

Lemma 1. *A state vector $x = (x_1, x_2, \ldots, x_m)$ is a d-MP candidate when its corresponding flow vector $f = (f_1, f_2, \ldots, f_p)$ satisfies the following conditions:*

$$f_1 + f_2 + \cdots + f_p = d, \tag{3}$$

$$0 \leq f_j \leq U_j, \quad \text{for } 1 \leq j \leq p, \tag{4}$$

$$0 \leq \sum_{e_i \in P_j} f_j \leq W_i, \quad \text{for } 1 \leq i \leq m, \tag{5}$$

$$x_i = \sum_{e_i \in P_j} f_j, \quad \text{for } 1 \leq i \leq m, \tag{6}$$

where $U_i = \min\{W_i | e_i \in P_j\}$ is the maximal capacity of P_j $(1 \leq j \leq p)$. Since each unit of flow sent from source node s to sink node t should travel through one of the MPs, the summation of flows travelling through all MPs must be equal to d, which is reflected by condition (3). Condition (4) points out that the flow travelling through P_j should not exceed the maximal capacity of P_j $(1 \leq j \leq p)$, and condition (5) indicates that the flow through e_i should not be above the largest capacity of e_i $(1 \leq i \leq m)$. Equation (6) pinpoints the relationship between the current state of e_i and the flow through e_i $(1 \leq i \leq m)$. Generally, the algorithms [22, 26–30] grounded on Lemma 1 consist of three steps:

Step 1: solve all d-MP candidates

Step 2: verify d-MP candidates to obtain real d-MPs

Step 3: remove duplicate d-MPs.

Each d-MP is also a d-MP candidate, and most of the existing methods need to search for all d-MP candidates prior to determining all d-MPs (Step 1). The feasible solutions derived from Lemma 1 are d-MP candidates. A d-MP candidate is not necessarily a d-MP; then, a verification step is required (Step 2). The methods for verifying d-MP candidates include the direct verification method [30], the comparison method [22, 26, 28], and the cycle-checking method [27]. The direct verification method is mainly based on the definition of d-MPs, that is, every d-MP candidate x derived from Lemma 1 is verified to determine whether it satisfies $M(x - 0(e_i)) < d$ for each $x_i > 0$. If the answer is "yes," x is a d-MP; otherwise, x is not a d-MP. The rationale behind the comparison method is that, for a d-MP candidate x, if there exists no d-MP candidate y such that $x > y$ ($x > y$ means $x_i \geq y_i$ for $i = 1, 2, \ldots, m$ and $x_j \geq y_j$ for at least one j $(1 \leq j \leq m)$), then x is a d-MP. A major drawback for the

comparison method is the high-computational complexity. The cycle-checking method is to check whether the network under x has a cycle. In contrast to the direct verification method and the comparison method, the cycle-checking method holds the efficiency advantage, but it is more applicable to directed networks. The set of d-MPs derived from Step 2 may contain duplicate d-MPs which significantly contribute to the computational burden of reliability evaluation but have no effect on the final reliability value, and then there is need to remove duplicate d-MPs (STEP 3).

3. The Proposed Algorithm

3.1. An Improved Model. A flow vector $f = (f_1, f_2, \ldots, f_p)$ satisfying conditions (3)–(5) is said to be a feasible flow vector. It is obvious that solving all of the feasible flow vectors is the first step and also the major obstacle for finding all d-MP candidates. The burden of solving feasible flow vectors is greatly dependent on the capacity ranges determined by conditions (4) and (5); thus, the solution efficiency would be improved if the capacity ranges in conditions (4) and (5) can be shrunk. As can be seen below, the lower bounds in condition (5) have the potential to be raised and the upper bounds in conditions (4) and (5) have the potential to be dropped.

Now, we define a special state vector $W(0_i) = (W_1, W_2, \ldots, W_{i-1}, 0, W_{i+1}, \ldots, W_m)$ in which capacity level is 0 for e_i and capacity level is the largest capacity for other edges; then, a crucial necessary condition for a state vector to be a d-MP can be attained.

Theorem 1. *For a state vector $x = (x_1, x_2, \ldots, x_m)$, if it is a d-MP, then $x_i \geq L_i$ for $1 \leq i \leq m$, where $L_i = \max\{d - M(W(0_i)), 0\}$.*

Proof. By definition, $L_i \geq 0$ holds. If $L_i = 0$, it is easy to have $x_i \geq L_i$.

If $L_i \geq 0$, it implies $L_i = d - M(W(0_i)) > 0$. Given that $M(W) \geq d$ holds, if $d - M(W(0_i)) > 0$, i.e., $M(W(0_i)) < d$, it means that at least $d - M(W(0_i))$ units of flow must travel through e_i in order for d units of flow to be transmitted from s to t. If x is a d-MP, then $M(x) = d$ follows from the definition, i.e., d units of flow can be transmitted from s to t under x. As a result, it is deduced that $x_i \geq d - M(W(0_i)) = L_i$, which completes the proof.

In fact, Theorem 1 reveals that $L_i\,(1 \leq i \leq m)$ is an improved lower capacity bound of e_i in d-MPs, and thus can be employed in condition (5) to shorten the capacity range. By the renowned max-flow algorithm, computing $M(W(0_i))$ requires $O(mn\log n)$ time [36]; then, the time complexity of determining all L_i for $1 \leq i \leq m$ is $O(m^2 n\log n)$. □

Corollary 1. *The time complexity of finding all L_i for $1 \leq i \leq m$ is $O(m^2 n\log n)$.*

Note that $f_1 + f_2 + \cdots + f_p = d$ in condition (3) means $f_j \leq d$ for $1 \leq i \leq m$. Also, it is clear that $\sum_{e_i \in P_j} f_j \leq f_1 + f_2 + \cdots + f_p = d$ holds. Hence, the upper bounds in conditions (4) and (5) can be replaced by $\min\{U_j, d\}$ and $\min\{W_i, d\}$, respectively, that is, the following improved model can be constructed to solve all d-MP candidates.

Theorem 2. *A state vector $x = (x_1, x_2, \ldots, x_m)$ is a d-MP candidate when its corresponding flow vector $f = (f_1, f_2, \ldots, f_n)$ satisfies the following conditions:*

$$f_1 + f_2 + \cdots + f_p = d, \tag{7}$$

$$0 \leq f_j \leq \min\{U_j, d\}, \quad \text{for } 1 \leq j \leq p, \tag{8}$$

$$L_i \leq \sum_{e_i \in P_j} f_j \leq \min\{W_i, d\}, \quad \text{for } 1 \leq i \leq m, \tag{9}$$

$$x_i = \sum_{e_i \in P_j} f_j, \quad \text{for } 1 \leq i \leq m. \tag{10}$$

Proof. Directly from Theorem 1 and Lemma 1.

It can be easily understandable that Theorem 2 is an improvement to Lemma 1 due to the stronger capacity constraints (8) and (9). Here, we adopt an example to illustrate the advantage of Theorem 2 in comparison with Lemma 1. The network in Figure 1 is a simple network with two MPs: $P_1 = \{e_1, e_2\}$ and $P_2 = \{e_3\}$, and the largest state vector is $W = (3, 2, 2)$. Given the demand level $d = 3$, a 3-MP candidate $x = (x_1, x_2, x_3)$, by Lemma 1, should satisfy the following conditions:

$$f_1 + f_2 = 3, \tag{11}$$

$$0 \leq f_1 \leq 2, \tag{12}$$

$$0 \leq f_2 \leq 2, \tag{13}$$

$$0 \leq x_1 = f_1 \leq 3, \tag{14}$$

$$0 \leq x_2 = f_1 \leq 2, \tag{15}$$

$$0 \leq x_3 = f_2 \leq 2, \tag{16}$$

that is,

$$f_1 + f_2 = 3, \tag{17}$$

$$0 \leq f_1 \leq 2, \tag{18}$$

$$0 \leq f_2 \leq 2, \tag{19}$$

$$x_1 = f_1, \tag{20}$$

$$x_2 = f_1, \tag{21}$$

$$x_3 = f_2. \tag{22}$$

Then, a total number of 9 flow vectors determined by conditions (18) and (19) need to be checked to solve 3-MP candidates in the worst case. When $d = 3$, $L_1 = \max$

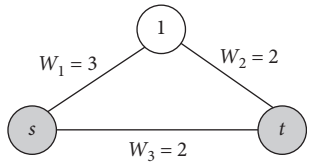

Figure 1: A simple network.

$\{0, d - M(W(0_1))\} = \max\{0, 3-2\} = 1$; similarly, $L_2 = 1$; $L_3 = 1$; $\min\{U_j, d\} = 2$ for $j = 1, 2$; $\min\{W_1, d\} = 3$ and $\min\{W_2, d\} = \min\{W_3, d\} = 2$. Thus, a 3-MP candidate $x = (x_1, x_2, x_3)$, by Theorem 2, should satisfy the following conditions:

$$\begin{aligned} f_1 + f_2 &= 3, \\ 0 \le f_1 &\le 2, \\ 0 \le f_2 &\le 2, \\ 1 \le x_1 = f_1 &\le 3, \\ 1 \le x_2 = f_1 &\le 2, \\ 1 \le x_3 = f_2 &\le 2, \end{aligned} \tag{23}$$

that is,

$$f_1 + f_2 = 3, \tag{24}$$

$$1 \le f_1 \le 2, \tag{25}$$

$$1 \le f_2 \le 2, \tag{26}$$

$$x_1 = f_1, \tag{27}$$

$$x_2 = f_1, \tag{28}$$

$$x_3 = f_2. \tag{29}$$

As a result, a total number of 4 flow vectors determined by conditions (25) and (26) need to be checked to solve 3-MP candidates in the worst case. By comparison, it can be seen that Theorem 2 holds an advantage over Lemma 1 owing to a smaller number of flow vectors to be checked. □

3.2. Removing Duplicate d-MPs. The set of d-MPs derived from Lemma 1 or Theorem 2 may contain duplicate d-MPs in the sense that one d-MP has the potential to be generated multiple times. Removing duplicate d-MPs is the most time-consuming step of the d-MP problem [22, 26–30]. The traditional method for removing duplicate d-MPs is the component-wise comparison method. Although the component-wise comparison method is simple and easy to understand, it has the deficiency of poor computational efficiency. In this section, an efficient technique will be developed to remove duplicate d-MPs and its computational efficiency is higher than the component-wise comparison method. Given a state vector x, its associated value $\mathbf{\Phi}(x)$ is defined as follows:

$$\mathbf{\Phi}(x) = \log\left(\sum_{i=1}^{m} x_i \pi^{i-1}\right). \tag{30}$$

Then, the following theorem indicates that there exists a one-to-one relationship between the state vector x and its associated value $\mathbf{\Phi}(x)$.

Theorem 3. *For any two state vectors x and y, $\Phi(x) = \Phi(y)$ if and only if $x = y$.*

Proof.

(1) If $x = y$, then $\Phi(x) = \Phi(y)$ is directly from the definition of associated value.

(2) By definition, $\Phi(x) = \log(\sum_{i=1}^{m} x_i \pi^{i-1})$ and $\Phi(y) = \log(\sum_{i=1}^{m} y_i \pi^{i-1})$. If $\Phi(x) = \Phi(y)$, then one can have $\log(\sum_{i=1}^{m} x_i \pi^{i-1}) = \log(\sum_{i=1}^{m} y_i \pi^{i-1})$, i.e., $\sum_{i=1}^{m} x_i \pi^{i-1} = \sum_{i=1}^{m} y_i \pi^{i-1}$, that is, $\sum_{i=1}^{m} (x_i - y_i)\pi^{i-1} = 0$ holds. Notice that $\sum_{i=1}^{m} (x_i - y_i)\pi^{i-1} = 0$ is a polynomial equation with integer coefficient, and π is a transcendental number; it follows that $\sum_{i=1}^{m} (x_i - y_i)\pi^{i-1} = 0$ if and only if all of its coefficients are equal to 0. Thus, $x_i - y_i = 0$ for all $i = 1, 2, \ldots, m$ when $\sum_{i=1}^{m} (x_i - y_i)\pi^{i-1} = 0$ holds, which means $x = y$.

The aim of introducing the concept of the associated value is to transfer each state vector into a unique number. Based on the relationship between a state vector and its associated value, the task of removing duplicate d-MPs can be accomplished by sorting all of the associated values. For example, consider the network in Figure 1, and it is trivial to obtain that $x = (1, 1, 2)$ and $y = (2, 2, 1)$ are all the 3-MPs. By equation (30), $\Phi(x) = \log(1 \times \pi^0 + 1 \times \pi^1 + 2 \times \pi^2) = 3.173075$ and $\Phi(y) = \log(2 \times \pi^0 + 2 \times \pi^1 + 1 \times \pi^2) = 2.898824$. Because $\Phi(x) = 3.173075 \neq \Phi(y) = 2.898824$, it is directly concluded that $x = (1, 1, 2)$ and $y = (2, 2, 1)$ are not duplicates each other.

For a state vector x, it takes $O(m)$ time to calculate its associated value $\Phi(x)$. Suppose that σ is the total number of d-MP candidates; then, it requires $O(m\sigma)$ time to calculate all of the associated values and takes $O(\sigma \log \sigma)$ time to removing duplicates by using the quick sort or merge sort method. Therefore, the time complexity of removing duplicate d-MPs is $O(m\sigma) + O(\sigma \log \sigma) = O(\sigma \log \sigma)$. Note that the time complexity of the traditional comparison method is $O(m\sigma^2)$. Because $O(\sigma \log \sigma) \ll O(m\sigma^2)$, the method based on Theorem 3 is more efficient in removing duplicates. Therefore, the developed technique is a favorable alternative to remove duplicate d-MPs. □

3.3. An Algorithm for Finding d-MPs. Using the obtained results, we suggest an algorithm to solve all d-MPs below. Given that all MPs are known in advance, all d-MPs without duplicates can be obtained using the following steps.

Input: A multistate network $G(V, E, W)$ with demand level d.

Output: All d-MPs without duplicates.

Step 0. Calculate $\min\{U_j, d\}$ for $1 \le j \le p$, $\min\{W_i, d\}$ and $L_i = \max\{0, d - M(W(0_i))\}$ for $1 \le i \le m$, and let $\Omega = \varnothing$.

Step 1. Use the fast enumeration algorithm to solve all of the feasible solutions (i.e., feasible flow vectors) to conditions (7)–(9), and transform each feasible solution into its corresponding d-MP candidate by equation (10). Suppose $x^1, x^2, \ldots, x^\sigma$ are all the obtained d-MP candidates.

Step 2. For each d-MP candidate $x^j = (x_1^j, x_2^j, \ldots, x_m^j)$ $(1 \le j \le \sigma)$, if $M(x_i^j - 0(e_i)) < d$ for all $x_i^j > 0$, then calculate $\Phi(x^j)$ by equation (30), and let $\Omega = \Omega \cup \{\Phi(x^j)\}$.

Step 3. Use the quick sort or merge sort method to remove all duplicate elements in Ω, and the remaining elements correspond to all of the d-MPs without duplicates.

Step 0 is a preprocessing step for calculating lower and upper capacity bounds in conditions (8) and (9). In Step 1, all d-MP candidates are solved by using the enumeration method. In particular, we notice that the fast enumeration method reported by Chen [37] has been proven to be more efficient than the traditional enumeration method; thus, it is used to solve all of the feasible flow vectors in Step 1. The details regarding the fast enumeration method can be referred to Chen [37] and Chen and Lin [28]. All d-MP candidates are verified in Step 2, and the associated value $\Phi(x^j)$ corresponding to each d-MP x^j is calculated. Step 3 is to remove duplicate d-MPs.

The computational complexity of every step is analyzed as follows. Step 0 takes $O(mp)$ time and $O(m)$ time to compute all $\min\{U_j, d\}$ $(1 \le j \le p)$ and all $\min\{W_j, d\}$ $(1 \le i \le m)$, respectively. By Corollary 1, finding all $L_i = \max\{0, d - M(W(0_i))\}$ $(1 \le i \le m)$ requires $O(m^2 n \log n)$ time. Thus, the computational complexity of Step 0 is $O(mp) + O(m) + O(m^2 n \log n) = O(mp)$. As analyzed in Section 3.1, it takes $O(\prod_{i=1}^{\pi} q_k)$ time to generate all feasible flow vectors, where π is the number of groups of alternative orders arranged by the fast enumeration method and q_k is the total number of enumerations in the ith group. It requires $O(m\sigma p)$ to transform all feasible flow vectors into d-MP candidates, where σ is the number of all feasible flow vectors, i.e., the number of d-MP candidates. As a result, the computational complexity of Step 1 is $O(\prod_{i=1}^{\pi} q_k) + O(m\sigma p)$. Checking all d-MP candidates requires $O(m^2 n \log n)$ time, and it takes $O(m\sigma)$ time to calculate all associated values in the worst case; then, Step 2 totally takes $O(m^2 \sigma n \log n)$ time. Using the quick sort or merge sort method to remove duplicates in Ω requires $O(\sigma \log \sigma)$ time in the worst case, so the computational complexity of Step 3 is $O(\sigma \log \sigma)$.

Given that the method by Chen and Lin [28] is regarded as an efficient algorithm for finding d-MPs, we compare it with the proposed algorithm in terms of computational complexity. In addition, while the method by Lin and Chen [30] is a newly reported method and is also considered to be efficient in solving d-MPs, it does not include the step of removing duplicates. Therefore, it is unfair to compare it with the proposed algorithm from the perspective of computational complexity. However, we will compare both algorithms through numerical examples in Section 5 when the step of removing duplicates is incorporated into the method by Lin and Chen [30].

Recall that the method by Chen and Lin [28] is based on Lemma 1 to search for all d-MP candidates. Note that the role of Step 1 for solving all d-MP candidates in the proposed algorithm is identical with that of Steps 1 and 2 in the method by Chen and Lin [28]. The theoretical results in Section 3.2 indicate that the time complexity of finding d-MP candidates based on Theorem 1 is upper bounded by the one based on Lemma 1; therefore, it is easy to conclude that the proposed algorithm holds an advantage over Chen and Lin's method [28] in solving d-MP candidates.

Moreover, the role of Steps 2 and 3 in the proposed algorithm is equivalent to that of Step 3 in the method by Chen and Lin. Step 3 of Chen and Lin's method utilizes the component-wise comparison method to find out d-MPs and remove duplicate d-MPs at the same time, so its computational complexity is $O(m\sigma^2)$. And yet, the computational complexity of Steps 2 and 3 in the proposed algorithm is totally $O(m^2 \sigma n \log n) + O(\sigma \log \sigma) = O(m^2 \sigma n \log n)$. Because $O(m^2 \sigma n \log n) < O(m\sigma^2)$, the computational complexity of the proposed algorithm in finding out d-MPs and removing duplicate d-MPs is lower than that of Chen and Lin's method [28]. Note that even though one more Step 0 needs to be implemented by the proposed algorithm compared with the method of Chen and Lin [28], it has no impact on the computational complexity of the whole algorithm considering $O(mp) < O(m2\sigma n \log n) < O(m\sigma p)$. Therefore, the proposed algorithm has a distinct efficiency advantage in solving d-MPs.

4. An Illustrative Example

The classical bridge network shown in Figure 2 is adopted to demonstrate how the proposed algorithm works. The data of edges are presented in Table 1. The network in Figure 2 has 4 MPs from s to t: $P_1 = \{e_1, e_2\}$, $P_2 = \{e_1, e_3, e_5\}$, $P_3 = \{e_4, e_5\}$, and $P_4 = \{e_4, e_3, e_2\}$. Suppose the demand level is $d = 3$; then, all 3-MPs can be obtained using the following steps.

Step 0: $\min\{U_1, d\} = \min\{2, 3\} = 2$, similarly, $\min\{U_2, d\} = 1$, $\min\{U_3, d\} = 2$, $\min\{U_4, d\} = 1$, and $\min\{W_1, d\} = \min\{3, 3\} = 3$, similarly, $\min\{W_2, d\} = 2$, $\min\{W_3, d\} = 1$, $\min\{W_4, d\} = 2$, $\min\{W_5, d\} = 2$, and $L_1 = \max\{0, d - M(W(0_i))\} = \max\{0, 1\} = 1$, similarly, $L_2 = 1, L_3 = 0, L_4 = 0$, and $L_5 = 1$.

Step 1: use the fast enumeration method to solve all of the feasible solutions $f = (f_1, f_2, f_3, f_4)$ to the following conditions:

$$\begin{aligned} f_1 + f_2 + f_3 + f_4 &= 3, \\ 0 \le f_1 &\le 2, \\ 0 \le f_2 &\le 1, \\ 0 \le f_3 &\le 2, \end{aligned}$$

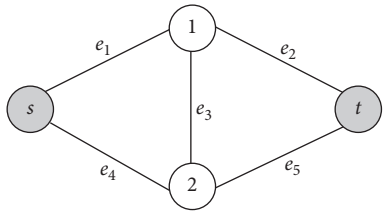

Figure 2: The bridge network.

Table 1: The data of edges for the bridge network in Figure 2.

Edge	States				State probabilities			
e_1	0	1	2	3	0.002	0.013	0.125	0.860
e_2	0	1	2	—	0.005	0.010	0.985	—
e_3	0	1	—	—	0.110	0.890	—	—
e_4	0	1	2	—	0.003	0.012	0.985	—
e_5	0	1	2	—	0.006	0.015	0.979	—

$$\begin{aligned} 0 \le f_4 \le 1, \\ 1 \le f_1 + f_2 \le 3, \\ 1 \le f_1 + f_4 \le 2, \\ 0 \le f_2 + f_4 \le 1, \\ 0 \le f_3 + f_4 \le 2, \\ 1 \le f_2 + f_3 \le 2. \end{aligned} \tag{31}$$

The derived feasible solutions are $f^1 = (1, 0, 2, 0)$, $f^2 = (1, 0, 1, 1)$, $f^3 = (1, 1, 1, 0)$, $f^4 = (2, 0, 1, 0)$, and $f^5 = (2, 1, 0, 0)$. By equation (11), Transform $f^j (1 \le j \le 5)$ into its corresponding d-MP candidate: $f^1 \longrightarrow x^1 = (1, 1, 0, 2, 2)$, $f^2 \longrightarrow x^2 = (1, 2, 1, 2, 1)$, $f^3 \longrightarrow x^3 = (2, 1, 1, 1, 2)$, $f^4 \longrightarrow x^4 = (2, 2, 0, 1, 1)$, and $f^5 \longrightarrow x^5 = (3, 2, 1, 0, 1)$.

Step 2: for $x^1 = (1, 1, 0, 2, 2), M(x^1 - 0(e_1)) = 2 < 3, M(x^1 - 0(e_2)) = 2 < 3, M(x^1 - 0(e_3)) = 2 < 3, M(x^1 - 0(e_4)) = 2 < 3$, and $M(x^1 - 0(e_5)) = 2 < 3$, then $\Phi(x^1) = 5.564414$, and $\Omega = \Omega \cup \{5.564414\}$. Similarly, for $x^2 = (1, 2, 1, 2, 1), M(x^2 - 0(e_i)) = 2 < 3$ for all $x_i^2 > 0$, then $\Phi(x^2) = 5.173743$ and $\Omega = \Omega \cup \{5.173743\}$; for $x^3 = (2, 1, 1, 1, 2), M(x^3 - 0(e_i)) = 2 < 3$ for all $x_i^3 > 0$, then $\Phi(x^3) = 5.484115$ and $\Omega = \Omega \cup \{5.484115\}$; for $x^4 = (2, 2, 0, 1, 1), M(x^4 - 0(e_i)) = 2 < 3$ for all $x_i^4 > 0$, then $\Phi(x^4) = 4.917778$ and $\Omega = \Omega \cup \{4.917778\}$; for $x^5 = (3, 2, 1, 0, 1), M(x^5 - 0(e_i)) = 2 < 3$, for all $x_i^5 > 0$, then $\Phi(x^5) = 4.758422$ and $\Omega = \Omega \cup \{4.758422\}$.

Step 3: there is no duplicate in Ω, then the elements in Ω correspond to all of the 3-MPs without duplicates: $x^1 = (1, 1, 0, 2, 2), x^2 = (1, 2, 1, 2, 1), x^3 = (2, 1, 1, 1, 2), x^4 = (2, 2, 0, 1, 1)$, and $x^5 = (3, 2, 1, 0, 1)$.

The solution results are summarized in Table 2, indicating there are totally five 3-MPs generated. Using the Inclusion-Exclusion method shown in Section 2.2, the final network reliability for this example is calculated as $R_3 = 0.986002$.

5. Efficiency Investigation by Numerical Examples

In this section, we investigate the efficiency of the proposed algorithm. As mentioned in Section 2.2, Lemma 1 proposed by Lin et al. [22] is one of the most important models with respect to d-MPs, and is also the foundation of most of the existing methods [26–30]. As an improvement to Lemma 1, Theorem 2 is the foundation of the proposed algorithm. So, we compare the proposed algorithm with the ones based on Lemma 1. Also, given that the methods by Chen and Lin [28] and Lin and Chen [30] are newly developed algorithms and also considered to be efficient in solving d-MPs, we implement the numerical experiments to compare the proposed algorithm with them. Meanwhile, since Lin and Chen [30] did not consider the step of removing duplicate d-MPs, we incorporate the proposed technique into the method of Lin and Chen for a fair comparison. All of the three algorithms require MPs as prior knowledge. Actually, MPs can be easily determined using the existing methods, such as the methods by Chen and Lin [38] and Bai et al. [39], so it is unnecessary to include this procedure when we compare the efficiency. All algorithms are coded in MATLAB 2009 and are implemented on a PC with Intel(R) Core (TM) i5-3210M 2.50 GHz CPU and 8 GB of RAM.

5.1. Example 1. First, we consider the network shown in Figure 3. The largest state vector is set to $W = (7, 5, 6, 4, 4, 5, 4, 6, 7)$, and the max-flow of the network under the largest state vector W is $D = 12$. To draw a full picture of the efficiency of different algorithms, all of the demand levels from 1 to 12 are examined, i.e., 1-MPs, 2-MPs, . . ., and 12-MPs are solved. In numerical experiments, we are interested in the required computational time for finding all d-MPs. In addition, we use ratios, defined as the CPU time consumed by the reported methods by Chen and Lin [28] and Lin and Chen [30] divided by the CPU time consumed by the proposed algorithm, respectively, to exhibit the relative efficiency for the sake of intuitively capturing the performances of different algorithms. The computational results are summarized in Table 3, including the number of d-MPs, the CPU times in seconds consumed by different algorithms and their ratios. By Table 3, we make the following observations.

Compared with the method by Chen and Lin [28], the proposed algorithm spends a smaller amount of CPU time in solving d-MPs for every demand level d. Thus, the proposed algorithm is more efficient than Chen and Lin's method. Notice that the ratio T_{CL}/T, representing the advantage of proposed algorithm over the method by Chen and Lin for finding d-MPs, is greater than 4 for every demand level d except $d = 12$, indicating that the proposed algorithm holds a distinct advantage in most cases. Furthermore, when the demand level d is intermediate ($d = 5, 6, 7, 8$), the ratio T_{CL}/T is above 30, which exhibits the considerable advantage of the proposed algorithm.

The CPU time consumed by the proposed algorithm is less than that consumed by Lin and Chen's method [30] in

Table 2: Solution results of the proposed algorithm.

j	Feasible solution f^j	The corresponding 3-MP candidate	A 3-MP?	$\Phi(x^j)$	A duplicate 3-MP
1	$f^1 = (1, 0, 2, 0)$	$x^1 = (1, 1, 0, 2, 2)$	Yes	5.564414	No
2	$f^2 = (1, 0, 1, 1)$	$x^2 = (1, 2, 1, 2, 1)$	Yes	5.173743	No
3	$f^3 = (1, 1, 1, 0)$	$x^3 = (2, 1, 1, 1, 2)$	Yes	5.484115	No
4	$f^4 = (2, 0, 1, 0)$	$x^4 = (2, 2, 0, 1, 1)$	Yes	4.917778	No
5	$f^5 = (2, 1, 0, 0)$	$x^5 = (3, 2, 1, 0, 1)$	Yes	4.758422	No

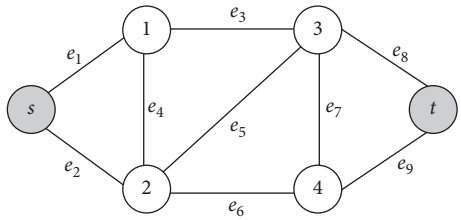

Figure 3: A test network for example 1.

Table 3: Results of different algorithms for solving the network in Figure 3.

d	# of d-MPs	Computational time (s)			Ratio	
		T_{CL}	T_{LC}	T	T_{CL}/T	T_{LC}/T
1	13	0.371	0.372	0.065	5.708	5.723
2	59	0.457	0.516	0.096	4.760	5.375
3	175	1.682	1.334	0.254	6.622	5.252
4	410	13.903	4.639	0.951	14.619	4.878
5	719	88.871	14.061	2.916	30.477	4.822
6	916	339.614	34.474	7.367	46.099	4.680
7	894	774.969	67.595	15.708	49.336	4.303
8	632	845.703	91.782	22.162	38.160	4.141
9	389	460.938	82.058	18.496	24.921	4.437
10	200	124.697	47.218	9.453	13.191	4.995
11	79	18.324	19.665	3.792	4.832	5.186
12	19	1.241	4.661	0.985	1.260	4.732

Note: T_{CL} and T_{LC} are the computational times of Chen and Lin's algorithm [28] and Lin and Chen's algorithm [30], respectively, and T is the computational time of the proposed algorithm.

terms of solving d-MPs for every demand level d, showing the proposed algorithm outperforms the method by Lin and Chen as well. It is noted that the ratio T_{LC}/T, representing the advantage of proposed algorithm over Lin and Chen's method for finding d-MPs, is between 4 and 6 for every demand level d, thereby the proposed algorithm has a distinct advantage in all cases.

5.2. Example 2. In this section, we consider a relatively larger network in Figure 4 and consider two scenarios, where the largest states of all edges are equal to 4 and 5, respectively. For each scenario, five demand levels, i.e., $d = D/2 - 2$, $D/2 - 1$, $D/2$, $D/2 + 1$, and $D/2 + 2$, where D is the max-flow of the network under the largest state vector, are examined. We still use ratios, defined as the CPU time consumed by the reported methods by Chen and Lin [28] and Lin and Chen [30] divided by the CPU time consumed by the proposed

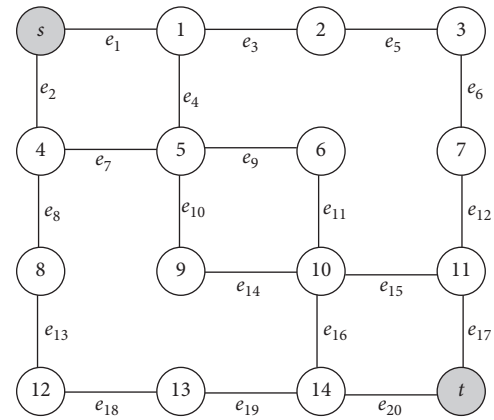

Figure 4: A test network for example 2.

algorithm, respectively, to exhibit the relative efficiency. The experimental results are summarized in Table 4, including the number of d-MPs, the CPU times in seconds consumed by different algorithms and their ratios.

It can be seen from Table 4 that all ratios (T_{CL}/T and T_{LC}/T) that denote the advantage of proposed algorithm over the methods by Chen and Lin [28] and Lin and Chen [30] for finding d-MPs are greater than 1 for every demand level d ($d = D/2 - 2, D/2 - 1, D/2, D/2 + 1, D/2 + 2$) in both scenarios; then, the proposed algorithm is more efficient than both methods. It is noteworthy that the method by Chen and Lin cannot find all 6-MPs (7-MPs) within 20000 CPU seconds, while the proposed algorithm requires 815.486 CPU seconds (1538.119 CPU seconds) that is far less than 20000 CPU seconds to generate all 6-MPs (7-MPs). In addition, the efficiency difference between the proposed algorithm and the method by Chen and Lin [28] is more prominent than that between the proposed algorithm and the method by Lin and Chen [30].

As an NP-hard problem, searching for d-MPs is a challenging task. The comparative results derived from Table 4 reveals that the proposed algorithm does contribute to the efficiency improvement of solving d-MPs without duplicates. Thus, the suggested algorithm can be used to accelerate the computation of network reliability. Network reliability is a key performance parameter in network design and improvement [9]. By employing the proposed algorithm, the network supervisor can efficiently capture the quality of service of real-world systems to make managerial decisions, especially to figure out the bottleneck of network improvement in many enterprises or organizations.

Table 4: Results of different algorithms for solving the network in Figure 4.

Scenario	d	# of d-MPs	Computational time (s)			Ratio	
			T_{CL}	T_{LC}	T	T_{CL}/T	T_{LC}/T
$W_i = 4$ $D = 8$	2	112	6.131	6.869	1.435	4.272	4.787
	3	392	15.685	14.926	11.183	1.403	1.335
	4	1057	215.084	67.208	61.763	3.482	1.088
	5	904	1104.979	258.804	172.544	6.404	1.500
	6	634	1983.916	545.056	267.265	7.423	2.039
$W_i = 5$ $D = 10$	3	392	64.246	62.902	11.261	5.705	5.586
	4	1057	264.896	115.446	61.791	4.287	1.868
	5	2408	4875.036	455.861	278.652	17.495	1.636
	6	2182	NA[a]	2320.584	815.486	—	2.846
	7	1736	NA[a]	8070.502	1538.119	—	5.247

[a]Not completed within 20000 CPU seconds.

6. Concluding Remarks

Reliability evaluation is an effective manner to capture the operational state of practical networks. One important type of method for multistate network reliability evaluation is using d-MPs. This paper presents a new efficient method to search for d-MPs for the multistate network reliability problem. On the one hand, an improved model is constructed to solve d-MPs by redefining capacity constraints of network components and MPs, and the model is an improvement to the well-known fundamental model. On the other hand, a technique superior to the traditional comparison method is proposed to remove duplicate d-MPs. The computational complexity of the proposed algorithm is analyzed, and a simple example is provided to illustrate the proposed algorithm. In addition, through computational experiments, it is found that the proposed algorithm is more efficient in terms of finding all d-MPs.

For future research, there is still potential for improving the suggested algorithm. For instance, transmission cost is also an important attribute of real-world networks, such as transportation networks or power transmission networks, and thus, extending the suggested algorithm to network reliability subject to cost constraint is worthy of study. Furthermore, it is interesting to study how to modify the suggested algorithm to be an approximate algorithm for tradeoff between execution time and the obtained set of d-MPs.

Notations

$G(V, E, W)$: A multistate network with a set of nodes $V = \{s, 1, 2, \ldots, n, t\}$, a set of edges $E = \{e_1, e_2, \ldots, e_m\}$, and the largest state vector $W = (W_1, W_2, \ldots, W_m)$
s/t: The source node/the sink node
m/n: The number of edges/the number of nodes except s and t
e_i: ith edge in E
x_i: A state of e_i
x: A state vector
$M(x)$: The max-flow of a network under x
d: Demand level
D: The max-flow of a network under W
R_d: Network reliability at demand level d
$0(e_i)$: A special state vector with the state of e_i being 1 and the states of other edges being 0, i.e., $0(e_i) = (0, \ldots, 0, 1, 0, \ldots, 0)$
$W(0_i)$: A special state vector $W(0_i) = (W_1, W_2, \ldots, W_{i-1}, 0, W_{i+1}, \ldots, W_m)$
L_i: $L_i = \max\{d - M(W(0_i)), 0\}$
f_j: The flow travelling through P_j
f: A flow vector $f = (f_1, f_2, \ldots, f_p)$
p: The number of MPs
P_i: The ith MP
U_j: $U_j = \min\{W_i \mid e_i \in P_j\}$ is the maximal capacity of P_j
$\Phi(x)$: The associated value with a state vector x
Ω: The set of $\Phi(x)$
π: The number of groups of alternative orders arranged by the fast enumeration method
q_k: The number of enumerations in the ith group
σ: The number of d-MP candidates.

Data Availability

All data are provided in full in the results section of this paper.

Conflicts of Interest

No conflicts of interest exist in this submission.

Acknowledgments

This work was mainly supported by the National Natural Science Foundation of China (Grant no. 71601072), Doctoral Project of Chongqing Federation of Social Science Circles (Grant no. 2019BS064), Planning Project of Human Social Science of Chongqing Municipal Education Commission (Grant nos. 20SKGH069 and 20SKGH063), and Scientific and Technological Research Program of Chongqing Municipal Education Commission (Grant nos. KJQN201900634 and KJQN201900649).

References

[1] Y.-K. Lin and C.-F. Huang, "System reliability of assured accuracy rate for multi-state computer networks from service level agreements viewpoint," *Journal of Systems Science and Systems Engineering*, vol. 23, no. 2, pp. 196–211, 2014.

[2] W.-C. Yeh, "Evaluation of the one-to-all-target-subsets reliability of a novel deterioration-effect acyclic multi-state information network," *Reliability Engineering & System Safety*, vol. 166, pp. 132–137, 2017.

[3] P.-C. Chang, Y.-K. Lin, and J. C. Chen, "System reliability for a multi-state manufacturing network with joint buffer stations," *Journal of Manufacturing Systems*, vol. 42, pp. 170–178, 2017.

[4] C.-C. Jane, "Performance evaluation of logistics systems under cost and reliability considerations," *Transportation Research Part E: Logistics and Transportation Review*, vol. 47, no. 2, pp. 130–137, 2011.

[5] C.-C. Jane and Y.-W. Laih, "Evaluating cost and reliability integrated performance of stochastic logistics systems," *Naval Research Logistics (NRL)*, vol. 59, no. 7, pp. 577–586, 2012.

[6] Y. F. Niu, W. H. K. Lam, and Z. Y. Gao, "An efficient algorithm for evaluating logistics network reliability subject to distribution cost," *Transportation Research Part E: Logistics and Transportation Review*, vol. 67, pp. 175–189, 2014.

[7] Y.-F. Niu, Z.-Y. Gao, and W. H. K. Lam, "Evaluating the reliability of a stochastic distribution network in terms of minimal cuts," *Transportation Research Part E: Logistics and Transportation Review*, vol. 100, pp. 75–97, 2017.

[8] A. Volkanovski, M. Čepin, and B. Mavko, "Application of the fault tree analysis for assessment of power system reliability," *Reliability Engineering & System Safety*, vol. 94, no. 6, pp. 1116–1127, 2009.

[9] E. Zio, "Reliability engineering: old problems and new challenges," *Reliability Engineering & System Safety*, vol. 94, no. 2, pp. 125–141, 2009.

[10] W.-C. Yeh and T.-C. Chu, "A novel multi-distribution multi-state flow network and its reliability optimization problem," *Reliability Engineering & System Safety*, vol. 176, pp. 209–217, 2018.

[11] G. Levitin, *The Universal Generating Function in Reliability Analysis and Optimization*, Springer-Verlag, London, UK, 2003.

[12] A. Lisnianski and G. Levitin, *Multi-State System Reliability: Assessment, Optimization and Applications*, World Scientific, Singapore, 2003.

[13] T. Aven, "Reliability evaluation of multistate systems with multistate components," *IEEE Transactions on Reliability*, vol. R-34, no. 5, pp. 473–479, 1985.

[14] Z. Hao, W.-C. Yeh, J. Wang, G.-G. Wang, and B. Sun, "A quick inclusion-exclusion technique," *Information Sciences*, vol. 486, pp. 20–30, 2019.

[15] G. Levitin and A. Lisnianski, "Importance and sensitivity analysis of multi-state systems using the universal generating function method," *Reliability Engineering & System Safety*, vol. 65, no. 3, pp. 271–282, 1999.

[16] J. E. Ramirez-Marquez and D. W. Coit, "A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability," *Reliability Engineering & System Safety*, vol. 87, no. 2, pp. 253–264, 2005.

[17] C. C. Jane and Y. W. Laih, "A practical algorithm for computing multi-state two-terminal reliability," *IEEE Transactions on Reliability*, vol. 57, no. 2, pp. 295–302, 2008.

[18] M. J. Zuo, Z. Tian, and H.-Z. Huang, "An efficient method for reliability evaluation of multistate networks given all minimal path vectors," *IIE Transactions*, vol. 39, no. 8, pp. 811–817, 2007.

[19] G. Bai, M. J. Zuo, and Z. Tian, "Ordering heuristics for reliability evaluation of multistate networks," *IEEE Transactions on Reliability*, vol. 64, no. 3, pp. 1015–1023, 2015.

[20] W.-C. Yeh, "An improved sum-of-disjoint-products technique for symbolic multi-state flow network reliability," *IEEE Transactions on Reliability*, vol. 64, no. 4, pp. 1185–1193, 2015.

[21] G. Bai, T. Liu, Y.-a. Zhang, and J. Tao, "An improved method for reliability evaluation of two-terminal multistate networks based on state Space decomposition," *IEEE Transactions on Reliability*, pp. 1–12, 2020.

[22] J.-S. Lin, C.-C. Jane, and J. Yuan, "On reliability evaluation of a capacitated-flow network in terms of minimal pathsets," *Networks*, vol. 25, no. 3, pp. 131–138, 1995.

[23] W.-C. Yeh, "A novel method for the network reliability in terms of capacitated-minimum-paths without knowing minimum-paths in advance," *Journal of the Operational Research Society*, vol. 56, no. 10, pp. 1235–1240, 2005.

[24] X.-Z. Xu, Y.-F. Niu, and Q. Li, "Efficient enumeration of *d*-minimal paths in reliability evaluation of multistate networks," *Complexity*, vol. 2019, Article ID 4561845, 10 pages, 2019.

[25] Y. Niu, Z. Gao, and H. Sun, "An improved algorithm for solving all *d*-MPs in multi-state networks," *Journal of Systems Science and Systems Engineering*, vol. 26, no. 6, pp. 711–731, 2017.

[26] Y.-K. Lin, "A simple algorithm for reliability evaluation of a stochastic-flow network with node failure," *Computers & Operations Research*, vol. 28, no. 13, pp. 1277–1285, 2001.

[27] W.-C. Yeh, "A simple method to verify all *d*-minimal path candidates of a limited-flow network and its reliability," *The International Journal of Advanced Manufacturing Technology*, vol. 20, no. 1, pp. 77–81, 2002.

[28] S.-G. Chen and Y.-K. Lin, "Searching for *d*-MPs with fast enumeration," *Journal of Computational Science*, vol. 17, pp. 139–147, 2016.

[29] M. Forghani-Elahabad and L. H. Bonani, "Finding all the lower boundary points in a multistate two-terminal network," *IEEE Transactions on Reliability*, vol. 66, no. 3, pp. 677–688, 2017.

[30] Y.-K. Lin and S.-G. Chen, "A maximal flow method to search for *d*-MPs in stochastic-flow networks," *Journal of Computational Science*, vol. 22, pp. 119–125, 2017.

[31] S. Satitsatian and K. C. Kapur, "An algorithm for lower reliability bounds of multistate two-terminal networks," *IEEE Transactions on Reliability*, vol. 55, no. 2, pp. 199–206, 2006.

[32] J. E. Ramirez-Marquez, D. W. Coit, and M. Tortorella, "A generalized multistate-based path vector approach to multi-state two-terminal reliability," *IIE Transactions*, vol. 38, no. 6, pp. 477–488, 2006.

[33] G. Bai, M. J. Zuo, and Z. Tian, "Search for all *d*-MPs for all *d* levels in multistate two-terminal networks," *Reliability Engineering & System Safety*, vol. 142, pp. 300–309, 2015.

[34] W.-C. Yeh, "Fast algorithm for searching *d*-MPs for all possible *d*," *IEEE Transactions on Reliability*, vol. 67, no. 1, pp. 308–315, 2018.

[35] Y.-F. Niu, Z.-Y. Gao, and W. H. K. Lam, "A new efficient algorithm for finding all *d*-minimal cuts in multi-state networks," *Reliability Engineering & System Safety*, vol. 166, pp. 151–163, 2017.

[36] R. K. Ahuja, M. Kodialam, A. K. Mishra, and J. B. Orlin, "Computational investigations of maximum flow algorithms," *European Journal of Operational Research*, vol. 97, no. 3, pp. 509–542, 1997.

[37] S. G. Chen, "Efficiency improvement in explicit enumeration for integer programming problems," in *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management*, Bangkok, Thailand, December 2013.

[38] S. G. Chen and Y. K. Lin, "Search for all minimal paths in a general large flow network," *IEEE Transactions on Reliability*, vol. 61, no. 4, pp. 949–956, 2012.

[39] G. Bai, Z. Tian, and M. J. Zuo, "An improved algorithm for finding all minimal paths in a network," *Reliability Engineering & System Safety*, vol. 150, pp. 1–10, 2016.