Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 6908120, 13 pages
http://dx.doi.org/10.1155/2016/6908120

Research Article

Decision Diagram Based Symbolic Algorithm for Evaluating the Reliability of a Multistate Flow Network

Rongsheng Dong, Yangyang Zhu, Zhoubo Xu, and Fengying Li

Guangxi Key Laboratory of Trusted Software, School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China

Correspondence should be addressed to Rongsheng Dong; ccrsdong@guet.edu.cn

Received 11 July 2016; Revised 6 November 2016; Accepted 16 November 2016

Academic Editor: J.-C. Cortés

Evaluating the reliability of Multistate Flow Network (MFN) is an NP-hard problem. Ordered binary decision diagram (OBDD) or variants thereof, such as multivalued decision diagram (MDD), are compact and efficient data structures suitable for dealing with large-scale problems. Two symbolic algorithms for evaluating the reliability of MFN, MFN_OBDD and MFN_MDD, are proposed in this paper. In the algorithms, several operating functions are defined to prune the generated decision diagrams. Thereby the state space of capacity combinations is further compressed and the operational complexity of the decision diagrams is further reduced. Meanwhile, the related theoretical proofs and complexity analysis are carried out. Experimental results show the following: (1) compared to the existing decomposition algorithm, the proposed algorithms take less memory space and fewer loops. (2) The number of nodes and the number of variables of MDD generated in MFN_MDD algorithm are much smaller than those of OBDD built in the MFN_OBDD algorithm. (3) In two cases with the same number of arcs, the proposed algorithms are more suitable for calculating the reliability of sparse networks.

1. Introduction

With the development of science and technology, network models have been applied to many systems in various fields, such as computer networking, communication, transportation, gas and oil production, power transmission, and logistics [1–3]. In the real world, many systems/networks have been abstracted as multistate systems (MSSs) [4–6], multistate networks (MSNs) [7–9], or stochastic flow networks (SFNs) [10–12]. The components (transmission lines or nodes) of the MSN have several states (failure, working perfectly, and demotion work), so each state of the components is usually modeled to a certain capacity with a probability. The MSNs that satisfy the conservation law are denoted multistate flow networks (MFNs), which are more adaptive to real-word network models [3, 13–16].

Network failures can be disastrous and losses great, so network reliability evaluation has become a major prerequisite to ensure proper functioning of a network [3, 9]. Network reliability is defined as the probability that a required amount of flow (such as data, power, etc.) is transmitted successfully from source node s to sink node t [3]. The reliability evaluation of a network is NP-hard problem [9]. A lot of algorithms have been provided based on some assumptions, including inclusion and exclusion method [10–12], and sum of disjoint products method [13]. These algorithms can provide an exact value of reliability, but they are limited to the enumeration of the variable combinations. Some approximation algorithms, like the upper and lower bound method [2], Monte Carlo simulation [17], and so forth possess relatively high efficiency but cannot get an exact value. Decision diagrams (DDs) have the advantages of high compactness and operability and of being able to express variable combinations implicitly and are thus one of the more feasible strategies to alleviate the problem of combinatorial explosion. Moreover, DDs can obtain an exact result with Shannon expansion or the extended form. Therefore, the decision diagram has been frequently used to evaluate network reliability because of the aforementioned advantages [18–24]. These algorithms are mainly for evaluating the reliability of the networks with two-state component [18] or a single path transmission [19–23].

For a multistate flow network, most existing reliability evaluation algorithms are based on the set of minimal cuts/paths [24, 25], and generating the set of minimal cuts/paths is another NP-hard problem [1, 26]. Jane and Laih have studied the reliability computation of two-terminal MFNs using a decomposition algorithm, which is better than the exhaustive enumeration method with a lower space complexity, with the prior requirement that the state vectors (d_flows) satisfy the demand without minimal cuts/paths [1, 7]. For the case of maximum flow and minimum cost, Terruggia and Bobbio presented a multivalued decision diagram (MDD) based method, which takes full advantage of the operation technology of multivalued decision diagrams so that the reliability evaluation of a MFN is simple and efficient, with a set of minimal cuts [24].

Thus, for calculating the reliability of a MFN, we attempt to propose an exact algorithm that simultaneously solves the following problems: (1) avoid searching for all minimal cuts/paths, and (2) alleviate enumerating the state space as far as possible. In view of this, first, we extend the formal definition of the MFN model to implement symbolic algorithms in this paper, and next two decision diagram based exact algorithms, MFN_OBDD and MFN_MDD, are proposed. A state vector (flow(d)) that satisfies the flow requirement is obtained by using the maximum flow algorithm in a constructed virtual network in the proposed algorithms, without locating all minimal cuts/paths. In MFN_OBDD, the ordered binary decision diagram (OBDD) structure and its operation technology are introduced, a capacity/state of each arc is represented by a Boolean variable, and four operating functions are defined. The reliability of a MFN is then described and solved accurately. The MDD structure with multibranch and operation technology is introduced; each arc is denoted by a multivalue variable, and three operating functions are defined in MFN_MDD. The correctness and efficiency of the algorithm are verified by experiments. Compared to the spatial decomposition algorithm of Jane and Laih [1], the proposed algorithms, which lower the computational complexity, do not require enumerating and decomposing the state space of the components.

The remainder of this article is organized as follows. In Section 2, an extended formal MFN model and a definition of MFN reliability are provided. In Section 3, elementary knowledge of OBDD and MDD is reviewed. In Section 4, the OBDD-based algorithm, MFN_OBDD, and the MDD-based algorithm, MFN_MDD, are proposed, and both their computational and storage complexity are analyzed. In Section 5, computational experiments are presented, and comparisons between the proposed algorithms and the decomposition algorithm of Jane and Laih are made. Conclusions are finally drawn in Section 6.

2. Multistate Flow Network Model and Reliability

In this section, a two-terminal (one source and one sink) MFN is modeled. Let $G = (s, t, V, A, S, C, P, D, d)$ be a MFN where s is the source, t is the sink, $V = \{v_i \mid 1 \le i \le n\}$ with $s, t \in V$ is the set of vertexes, $A = \{a_i \mid 1 \le i \le n\}$ with $A \subset V \times V$ is the set of arcs, $C = \{\mathbf{C}_i \mid 1 \le i \le n\}$ with $\mathbf{C}_i = \{c_{i,0}, c_{i,1}, \ldots, c_{i,(s_i-1)}\}$ is the set of a_i's capacities, $S = \{s_i \mid 1 \le i \le n,\ s_i = |\mathbf{C}_i|\}$ is the set of capacity numbers, $P = \{\mathbf{P}_i \mid 1 \le i \le n\}$ with $\mathbf{P}_i = \{p_{i,0}, p_{i,1}, \ldots, p_{i,(s_i-1)}\}$ is the set of a_i's capacity probabilities, $p(c_{i,j}) = p_{i,j}$, $D = \{d_0, d_1, \ldots, d_k\}$ is the set of the output flows from s to t, obviously, $\max\{d_i\}$ is the maximum flow of the network, d is the demand transmitted from s to t, and $d \in D$, $0 \le d \le \max\{d_i\}$. In addition, the following mapping relationship was established:

$$\mathbf{C}_1 \times \mathbf{C}_2 \times \cdots \times \mathbf{C}_n \longrightarrow D. \tag{1}$$

An important property of MFN is that each node satisfies the flow conservation law. For a node, the total amount of traffic input is equal to the total output flow. Thus, the flow from the source into the network is equal to the output from the sink.

In this paper, network G must satisfy the following assumptions:

(1) Each vertex is perfectly reliable. For the unreliable vertex case, vertexes and arcs can be treated as components of the same type.

(2) The capacity of each arc is stochastic with a given probability distribution.

(3) The capacities of different arcs are stochastically independent.

The vector of capacities $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ denotes the state vector of the network where $c_i \le \mathbf{C}_i$ is the state of a_i. $F(\mathbf{c})$ denotes the maximum flow from s to t. Vector $\mathbf{c}$ is satisfiable for a specified demand d if $F(\mathbf{c}) \ge d$; otherwise it is unsatisfiable. A satisfiable flow(d) can be obtained by Lemma 1.

Lemma 1 (see [1]). *Given a MFN, $G = (s, t, V, A, S, C, P, D, d)$, and a satisfiable state vector flow(d) can be obtained at least when $\overline{\mathbf{c}}$ is satisfiable, where flow(d) $= (f_1^d, f_2^d, \ldots, f_n^d)$ with $F(\text{flow}(d)) = d$ consists of flows f_i^d through each arc a_i, and $\overline{\mathbf{c}} = (c_1^{\max}, c_2^{\max}, \ldots, c_n^{\max})$ with $c_i^{\max} = \max\{c_{i,0}, c_{i,1}, \ldots, c_{i,(s_i-1)}\}$ is the vector of the maximum capacities of arcs.*

Proof. Construct a new network G^* based on the network G with a fictitious vertex s^* and a fictitious arc a_0 that has a fixed capacity d from s^* to s. Obviously, $\{a_0\}$ is a minimal cut set of network G^*. According to the maximum-flow minimum-cut theorem, the maximum flow from s^* to t is d. Each arc of G^*, $a_i \in A, (1 \le i \le n)$, has a flow recorded as f_i^d when G^* obtains the maximum flow. These flows constitute a state vector as $(f_1^d, f_2^d, \ldots, f_n^d)$; that is, there is a feasible flow recorded as flow(d) $= (f_1^d, f_2^d, \ldots, f_n^d)$ to satisfy the demand d. Therefore, the lemma is proved. □

For a MFN, we deduce the conclusion from the above. If $\underline{\mathbf{c}}$ is satisfiable for demand d with $\underline{\mathbf{c}} = \text{flow}(d) = (f_1^d, f_2^d, \ldots, f_n^d)$, $F(\underline{\mathbf{c}}) = d$, and $F(\overline{\mathbf{c}}) \ge d$, then $F(\mathbf{c}) \ge d$ with $\mathbf{c} \in \Omega$ and $\Omega = \{(c_1, c_2, \ldots, c_n) \mid f_i^d \le c_i \le c_i^{\max},$

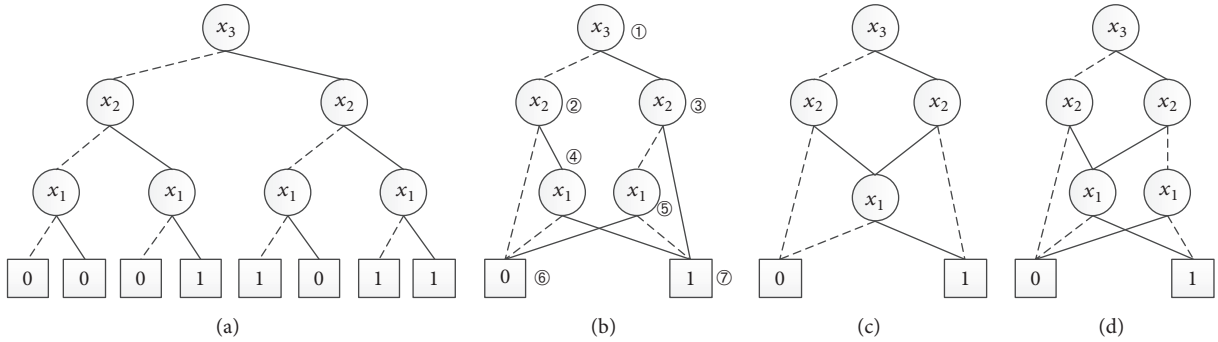

Figure 1: OBDD: (a) the complete binary tree of $W = x_1 \cdot x_2 + x_1' \cdot x_3$; (b) the OBDD of W; (c) the OBDD of U; (d) the OBDD of $W \cdot U$.

$c_i \in \mathbf{C}_i\} = [\underline{\mathbf{c}}, \overline{\mathbf{c}}]$. According to Corollary 1 of [1], a flow(d) may be obtained in $O(m^3)$ time.

In this paper, we study the reliability evolution of a MFN with a demand d. Then the MFN's reliability is defined as follows.

Definition 2. For a MFN G with demand d from source s to sink t, the probability of the state space corresponding to the satisfiable vectors of the network with d is the reliability of the MFN. That is, the reliability of G is

$$\begin{aligned} \mathrm{Rel}(d) &= P\{\mathbf{c} \mid F(\mathbf{c}) \geq d\} \\ &= P\{(c_1, c_2, \ldots, c_n) \mid F(\mathbf{c}) \geq d,\ c_i \in \mathbf{C}_i\} \\ &= \bigcup \{P(c_1, c_2, \ldots, c_n) \mid F(\mathbf{c}) \geq d,\ c_i \in \mathbf{C}_i\}. \end{aligned} \tag{2}$$

3. Ordered Binary Decision Diagram and Multivalued Decision Diagram

In this section, we review the concept of ordered binary decision diagrams (OBDD) and Multivalued Decision Diagram (MDD). An OBDD is a directed acyclic tree for Shannon decomposition and provides a compact, canonical, and efficiently manipulative representation for Boolean functions. A MDD is an extension of an OBDD.

3.1. Ordered Binary Decision Diagram. If F denotes a Boolean expression on $X = \{x_1, x_2, \ldots, x_n\}$ with x_i $(1 \leq i \leq n)$ being a Boolean variable of X, the **ite** (If-Then-Else) format of a binary decision diagram (BDD) is defined as

$$F = \mathbf{ite}(x, F_1, F_2) = x \cdot F_1 + x' \cdot F_2, \tag{3}$$

where $x \in X$, x' is the complement of x (if $x = 0$, $x' = 1$; else if $x = 1$, $x' = 0$), $F_1 = F_{x=1}$, and $F_2 = F_{x=0}$. A BDD has two terminal nodes that correspond to Boolean constants, 0 and 1, respectively. Each nonterminal node denotes a Boolean variable x with two branches corresponding to the Boolean expressions F_1 of $x = 1$ and F_2 of $x = 0$ for Shannon decomposition. Then each nonterminal node in BDD represents an **ite**(x, F_1, and F_2) and it is different from the others.

An ordered binary decision diagram (OBDD) is a BDD with a constant order for Boolean variables. The order of the Boolean variables from root to terminal is descending or increasing. Some operations can be defined between two OBDDs with the same variable order, and a new compact OBDD with the same variable order is obtained by an operation.

For example, Figure 1 shows the complete binary tree (Figure 1(a)) and the OBDD (Figure 1(b)) for the Boolean expression $W = x_1 \cdot x_2 + x_1' \cdot x_3$ with the variable order $\Pi : x_3 \succ x_2 \succ x_1$. Obviously, the OBDD stores the same information with fewer nodes. If we trace the path ① → ③ → ⑤ → ⑦ and reach terminal node 1, the value of $W = x_1 \cdot x_2 + x_1' \cdot x_3$ with variable assignment $(x_1, x_2, x_3) = (0, 0, 1)$ is 1. Figure 1(c) shows the OBDD for expression $U = x_1 \cdot x_2 + x_2' \cdot x_3$ with the same variable order, and Figure 1(d) shows the new OBDD obtained by an operation "$\cdot$" (conjunctive, **And**) between the two OBDDs.

3.2. Multivalued Decision Diagram. A multivalued discrete function F denotes the operation with n multivalued variables $\{x_1, x_2, \ldots, x_n\}$ mapping to the range $\mathbf{R}$.

$$\mathbf{F} : \mathbf{L}_1 \times \mathbf{L}_2 \times \cdots \times \mathbf{L}_n \longrightarrow \mathbf{R}, \tag{4}$$

where $\mathbf{L}_i = \{0, 1, \ldots, l_i - 1\}$ is the domain of variable x_i and $\mathbf{R} = \{r_0, r_1, \ldots, r_{k-1}\}$ is the range.

A MDD is a variant of a BDD. If F denotes an expression on $X = \{x_1, x_2, \ldots, x_n\}$, the **case** format of a MDD is defined as

$$\begin{aligned} F &= \mathbf{case}(x, F_{x=0}, F_{x=1}, \ldots, F_{x=l-1}) \\ &= (x = 0) \cdot F_{x=0} + (x = 1) \cdot F_{x=1} + \cdots + (x = l - 1) \\ &\quad \cdot F_{x=l-1} = \sum_{i=0}^{l-1} (x = i) \cdot F_{x=i}, \end{aligned} \tag{5}$$

where $x \in X$. A MDD has $k = |\mathbf{R}|$ terminal nodes that correspond to the constants, $\{r_0, r_1, \ldots, r_{k-1}\}$, respectively, and each nonterminal node denotes a variable x with $|\mathbf{D}_i|$ branches corresponding to the expressions $F_{x=i}$. Then each nonterminal node in a MDD represents a **case** format.

Similar to an OBDD, a new compact MDD is obtained by an operation between two different MDDs with the same

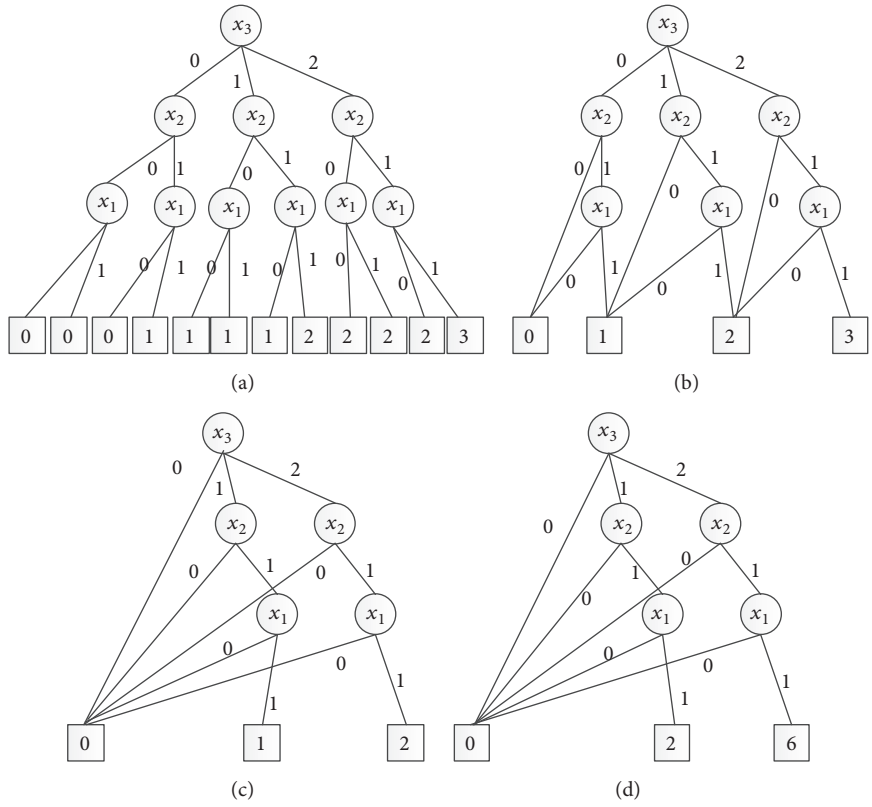

Figure 2: MDD: (a) the complete tree of $W = x_1 \cdot x_2 + x_3$; (b) the MDD of W; (c) the MDD of U; (d) the MDD of $W * U$.

variable order, and Property 1 is applicable for an OBDD or MDD. For example, Figure 2 shows the complete tree (Figure 2(a)) and the MDD (Figure 2(b)) for the numerical function $W = x_1 \cdot x_2 + x_3$ with the variable order $\Pi : x_3 \succ x_2 \succ x_1$, $\mathbf{L}_1 = \{0, 1\}$, $\mathbf{L}_2 = \{0, 1\}$, and $\mathbf{L}_3 = \{0, 1, 2\}$. Obviously, the MDD stores the same information with quite a few nodes. If we trace the path $(x_3 = 1) \to (x_2 = 1) \to (x_1 = 1) \to 2$ and reach terminal node 1, the value of $W = x_1 \cdot x_2 + x_3$ with variable assignment $(x_1, x_2, x_3) = (1, 1, 1)$ is 2. Figure 2(c) shows the MDD for the expression $U = x_1 \cdot x_2 \cdot x_3$ with the same variable order, and Figure 2(d) shows the new MDD obtained by an operation "$*$" (multiplication) between the two MDDs (if $W = 1$ and $U = 2$, then $W * U = 2$). In fact, the MDD can be further simplified. For $W * U$, if $W * U > 0$ satisfies the requirement, terminal nodes 2 and 6 can be combined in one node 1 (True). This process will be used in our algorithm.

Property 1. Applying the operation for a nonnull OBDD (or MDD) W and U will unite the intersecting spaces.

4. Symbolic Algorithms for MFN Reliability Evaluation

Because of the efficient data structures of decision diagrams, we take advantage of a decision diagram to evaluate the reliability of a MFN defined in Section 2. We present the MFN_OBDD algorithm based on OBDD in Section 4.1 and the MFN_MDD algorithm based on MDD in Section 4.2. An example of the proposed algorithms is given in Section 4.3 and the complexity analysis of the algorithms is provided in Section 4.4. It is worth declaring that we are not discussing the decision diagram variables order, because it is an independent NP-hard problem. In this paper, the variables order is determined by the indexes of the edges and the edge states.

4.1. OBDD-Based Algorithm MFN_OBDD for Reliability Evaluation. The OBDD is used to study two-state system because of the two branches of the nodes in an OBDD. In this section, we propose an OBDD-based algorithm, MFN_OBDD, for the reliability evaluation of a MFN. The method for variable encoding refers to Zang et al.'s algorithm [19]; that is, each state of an arc is represented by a Boolean variable.

The algorithm MFN_OBDD is comprised of two major parts: (1) generation of the OBDD that implicitly expresses all of the state vectors that satisfy the demand d, without listing the vectors; and (2) calculation of the reliability of the MFN by using the defined specific function and traversing the obtained OBDD. The steps of MFN_OBDD are performed as follows.

Step 1. Let each state of an arc be a Boolean variable. $A = \{a_1, a_2, \ldots, a_n\}$, and $C_i = \{c_{i,0}, c_{i,1}, \ldots, c_{i,(s_i-1)}\}$, for arc a_i, $\{a_{i,j} \mid 0 \le j \le s_i - 1\}$ denotes the set of state variables, and $c(a_{i,j}) = c_{i,j}$. According to the index of arcs, $a_n \succ \cdots \succ a_1 \succ a_2$, and the index of states of a_i, $a_{i,(s_i-1)} \cdots \succ a_{i,j} \cdots \succ a_{i,1} \succ a_{i,0}$, we

specify the variable order of the OBDDs to be $\Pi : a_{n,(s_n-1)} \succ \cdots a_{n,1} \succ a_{n,0} \cdots \succ a_{1,(s_1-1)} \succ \cdots \succ a_{1,0}$.

Step 2. According to Corollary 1 in [1], we can obtain a flow $\text{flow}(d)_1 = f_1^d(a_1, a_2, \ldots, a_2) = (f_{1,1}^d, f_{1,2}^d, \ldots, f_{1,n}^d)$; let $\underline{\mathbf{c}} = \text{flow}(d)_1$. The maximum capacities of arcs consist of the upper bound vector $\overline{\mathbf{c}} = (c_1^{\max}, c_2^{\max}, \ldots, c_n^{\max})$. Thus, all state vectors in the vector space $\Omega = \{(c_1, c_2, \ldots, c_n) \mid f_{1,i}^d \le c_i \le c_i^{\max}, c_i \in \mathbf{C}_i\} = [\underline{\mathbf{c}}, \overline{\mathbf{c}}]$ satisfy demand d. Then we generate the OBDD of Ω.

Step 2.1. Generate the BDD of $a_{i,j}$ with $c(a_{i,j}) \ge f_{1,i}^d$ with a defined operating function, named $a_{i,j}$_BDD. The generation process is based on the following rules in the function:

$$\begin{aligned} a_{i,j_{\text{BDD}}} &= \text{StateBDD}\left(a_{i,j}, c_{i,j}, f_{1,i}^d\right) \\ &= \begin{cases} 1, & a_{i,j} = 1, c_{i,j} \ge f_{1,i}^d, \\ 0, & \text{otherwise}, \end{cases} \\ & c_{i,j} \in \mathbf{C}_i,\ j \in (0, 1, \ldots, s_i - 1),\ i \in (1, 2, \ldots, n). \end{aligned} \tag{6}$$

Step 2.2. Construct the OBDD of arc a_i with $c(a_i) \ge f_{1,i}^d$, named a_i_OBDD, with a function as defined in the following formula:

$$\begin{aligned} a_i\text{\_OBDD} &= \text{ArcOBDD}\left(a_i, \right. \\ &\left. \left\{a_{i,1}\text{\_BDD}, a_{i,2}\text{\_BDD}, \ldots, a_{i,(s_i-1)}\text{\_BDD}\right\}\right) = a_{i,1}\text{\_BDD} \\ &+ a_{i,2}\text{\_BDD} + \cdots + a_{i,(s_i-1)}\text{\_BDD} \\ &= \begin{cases} 1, & \exists j, a_{i,j}\text{\_BDD.high} = 1, \\ 0, & \text{otherwise}, \end{cases} \\ & j \in (0, 1, \ldots, s_i - 1),\ i \in (1, 2, \ldots, n), \end{aligned} \tag{7}$$

where "+" between $a_{i,1}$_BDD and $a_{i,2}$_BDD denotes a basic operation of BDD, that is called **Or,** usually equivalent to disjunctive. And "$a_{i,j}$_BDD.high" represents the high branch.

Step 2.3. Use f_1^d_OBDD with a null initialization to save the OBDD of the state space Ω, and generate f_1^d_OBDD by a function as created in the following formula:

$$\begin{aligned} f_1^d\text{\_OBDD} &= \text{FlowOBDD}\left(a_1, a_2, \ldots, a_n\right) \\ &= a_1\text{\_OBDD} \cdot a_2\text{\_OBDD} \cdot \ldots \cdot a_n\text{\_OBDD} \\ &= \begin{cases} 1, & \forall i, a_i\text{\_OBDD} = 1, \\ 0, & \text{otherwise}, \end{cases} \\ & i \in (1, 2, \ldots, n), \end{aligned} \tag{8}$$

where "·" between a_1_OBDD and a_2_OBDD denotes a basic operation of BDD, that is called **And,** usually equivalent to conjunctive.

Step 3. The final OBDD is stored in d_OBDD, and the initialization is null. d_OBDD is generated with a function as the following:

$$\begin{aligned} d\text{\_OBDD} &= \text{DemandOBDD}\left(d\text{\_OBDD}, f_1^d\text{\_OBDD}\right) \\ &= d\text{\_OBDD} + f_1^d\text{\_OBDD} \\ &= \begin{cases} 1, & (d\text{\_OBDD} = 1) \vee \left(f_1^d\text{\_OBDD} = 1\right), \\ 0, & \text{otherwise}. \end{cases} \end{aligned} \tag{9}$$

Step 4. Loop to execute Steps 2 and 3 until all $\text{flow}(d)_k$ is searched. A new flow satisfying the demand d, $\text{flow}(d)_k = f_k^d(a_1, a_2, \ldots, a_n) = (f_{k,1}^d, f_{k,2}^d, \ldots, f_{k,n}^d)$, is obtained, and f_k^d_OBDD is generated under Π; then execute DemandOBDD() with replacing f_k^d_OBDD to f_1^d_OBDD in formula (9).

Step 5. Obtain the OBDD that implicitly expresses all the state vectors satisfying the demand d. Then construct a specific function, which is implemented with Lemma 3 and the Proof, with the state probability to traverse d_OBDD, and the reliability of G is obtained.

In algorithm MFN_OBDD above, Steps 1–4 generate the OBDD that implicitly expresses all the state vectors satisfying the demand d, and Step 5 calculates the MFN reliability.

Lemma 3. *If the* ***ite*** *operation of d_OBDD is defined as*

$$F = \textbf{ite}\left(a_{i,j}, U, W\right) = a_{i,j} \cdot U + a'_{i,j} \cdot W, \tag{10}$$

then the reliability can be calculated by a formula as

$$P(F = 1) = p_{i,j} \cdot P(U = 1) + P(W = 1), \tag{11}$$

where $p_{i,j}$ is the probability of $c_{i,j}$.

Proof. In algorithm MFN_OBDD, we note that $a_{i,j}$ is a Boolean variable that has two branches, but the probability of $c_{i,j}$ is fixed and unique. So we write a function

$$h\left(a_{i,j}\right) = \begin{cases} 1, & a_{i,j} = 0, \\ p_{i,j}, & a_{i,j} = 1, \end{cases} \tag{12}$$

which can be explained as a virtual probability distribution of $a_{i,j}$. Thus, the following deduction is produced:

$$\begin{aligned} P(F = 1) &= E\left(a_{i,j} \cdot U + a'_{i,j} \cdot W\right) \\ &= E\left(a_{i,j}\right) \cdot E(U = 1) + E\left(a'_{i,j}\right) \cdot E(W = 1) \\ &= h\left(a_{i,j} = 1\right) \cdot P(U = 1) + h\left(a_{i,j} = 0\right) \\ &\quad \cdot P(W = 1) \\ &= p_{i,j} \cdot P(U = 1) + P(W = 1). \end{aligned} \tag{13}$$

That is, we can calculate the reliability of MFN_OBDD with this deduction when d_OBDD is traversed; $\text{Rel}(d) = P(F = 1)$. Therefore, the lemma is proved. □

4.2. MDD-Based Algorithm MFN_MDD for Reliability Evaluation. A MDD with multibranches has often been used for reliability evaluation of multistate networks. Here, a MDD-based algorithm, MFN_MDD, is proposed for evaluating the MFN reliability. In algorithm MFN_MDD, each arc is represented by a multivalued variable.

The algorithm MFN_MDD consists mainly of two parts: (1) construction of the MDD that implicitly expresses all of the state vectors satisfying the demand d, without listing the vectors, and (2) obtaining the MFN reliability by traversing the MDD. The steps in executing algorithm MFN_MDD are as follows.

Step 1. Let each arc a_i be a MDD variable. The variable order is specified by $\Pi' : a_n \succ \cdots \succ a_2 \succ a_1$.

Step 2. It is the same as Step 2 in algorithm MFN_OBDD, and we obtain the flow$(d)_1$ and the space Ω. The MDD of Ω is generated following Steps 2.1 and 2.2.

Step 2.1. In order to simplify the format of MDD, generate the MDD with only two terminal nodes (0, 1) of arc a_i, named a_i_MDD, with a defined operation. The MDD is generated according to the following rules:

$$a_i\text{\_MDD} = \text{ArcMDD}\left(a_i, c_{i,j}, f_{1,i}^d\right) = \begin{cases} 1, & c_{i,j} \geq f_{1,i}^d, \\ 0, & c_{i,j} < f_{1,i}^d, \end{cases} \quad (14)$$

$$c_{i,j} \in \mathbf{C}_i,\ i \in (1, 2, \ldots, n),\ j \in (0, 1, \ldots, s_i - 1).$$

Step 2.2. According to the variable order Π', the MDD, named f_1^d_MDD, is constructed relying on the operation defined with following formula:

$$\begin{aligned} &f_1^d\text{\_MDD} \\ &\quad = \text{FlowMDD}\left(a_1\text{\_MDD}, a_2\text{\_MDD}, \ldots, a_n\text{\_MDD}\right) \\ &\quad = a_1\text{\_MDD} \cdot a_2\text{\_MDD} \cdot \ldots \cdot a_n\text{\_MDD} \\ &\quad = \begin{cases} 1, & \forall a_i\text{\_MDD.state}\,[j] = 1, \\ 0, & \text{otherwise}, \end{cases} \\ &\qquad\qquad i \in (1, 2, \ldots, n),\ j \in (0, 1, \ldots, s_i - 1), \end{aligned} \quad (15)$$

where "$\cdot$" between a_1_MDD and a_2_MDD denotes a basic operation of MDD, similar to BDD, that is called **And,** usually equivalent to conjunctive.

Step 3. Generate the result MDD, named d_MDD with a null initialization, with the operation function defined in the following formula:

$$\begin{aligned} d\text{\_MDD} &= \text{DemandMDD}\left(d\text{\_MDD}, f_1^d\text{\_MDD}\right) \\ &= d\text{\_MDD} + f_1^d\text{\_MDD} \\ &= \begin{cases} 1, & (d\text{\_MDD} = 1) \vee \left(f_1^d\text{\_MDD} = 1\right), \\ 0, & \text{otherwise}, \end{cases} \end{aligned} \quad (16)$$

where "+" between d_MDD and f_1^d_MDD denotes another basic operation of MDD, that is called **Or,** usually equivalent to disjunctive.

Step 4. Loop to execute Steps 2 and 3 until all flow$(d)_i$ is searched the same as in Step 4 in algorithm MFN_OBDD, and a flow $\text{flow}(d)_k = f_k^d(a_1, a_2, \ldots, a_n) = (f_{k,1}^d, f_{k,2}^d, \ldots, f_{k,n}^d)$ is obtained. Then f_k^d_MDD is generated under Π', and d_MDD is constructed by DemandOBDD() with replacing f_k^d_MDD with f_1^d_MDD in formula (16).

Step 5. At this point, d_MDD obtained can implicitly express all of the state vectors satisfying the demand d. Then, a specific function is constructed with Lemma 4 and the Proof with the state probability to traverse d_MDD, and the reliability of G is obtained.

Lemma 4. *If **case** of the d_MDD algorithm MFN_MDD is defined as*

$$F = \textbf{case}\left(a_i, F_{s(a_i)=0}, F_{s(a_i)=1}, \ldots, F_{s(a_i)=s_i-1}\right), \quad (17)$$

where $s(a_i) = j$ denotes that a_i is in the state of j with capacity $c_{i,j}$, then the reliability can be calculated by a formula as

$$P(F = 1) = \sum_{j=0}^{s_i-1} p_{i,j} \cdot P\left(F_{s(a_i)=j} = 1\right), \quad (18)$$

where $p_{i,j}$ is the probability of $c_{i,j}$.

Proof. In algorithm MFN_MDD, executing Step 1 obtains $\text{flow}(d)_k = (f_{k,1}^d, f_{k,2}^d, \ldots, f_{k,n}^d)$ satisfying demand d, and each state vector in $[\text{flow}(d)_i, \overline{\mathbf{c}}]$ is satisfiable. After executing Steps 2 and 3, d_MDD is obtained. All of the paths in d_MDD from root to terminal node 1 represent the disjoint state space for demand d, named $\Omega_{\text{flow}(d)} = \{\mathbf{c} \mid \mathbf{c} \in \bigcup_k [\text{flow}(d)_k, \overline{\mathbf{c}}]\} = \{\mathbf{c} \mid F(\mathbf{c}) \geq d\}$. Thus, according to Definition 2, we can calculate the reliability, $\text{Rel}(d) = P\{\mathbf{c} \mid F(\mathbf{c}) \geq d\}$, with the state probability $p_{i,j}$ to traverse d_MDD. Then,

$$\begin{aligned} P &= \{\mathbf{c} \mid F(\mathbf{c}) \geq d\} = P(F = 1) \\ &= \sum_{j=0}^{s_n-1} p\left(s\left(a_n\right) = j\right) \cdot P\left(F_{s(a_n)=j} = 1\right) \\ &= \sum_{j=0}^{s_n-1} p_{n,j} \cdot P\left(F_{s(a_n)=j} = 1\right), \end{aligned} \quad (19)$$

and because the root is a_n, we traverse d_MDD starting with a_n above. Thus, the exact reliability is obtained, and thus the lemma is proved. □

4.3. Illustrated Example. A multistate computer network described in [1, 7] is depicted in Figure 3 and is used to illustrate the proposed algorithms MFN_OBDD and MFN_MDD. The data of the arcs are shown in Table 1. Vertex 1 is the source, and vertex 4 is the sink. The demand from s to t reached 3; that is, $d = 3$.

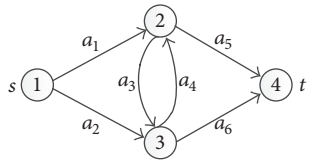

Figure 3: A multistate computer network.

Table 1: Data of arcs of Figure 3.

Arc	States/capacities				Probabilities			
	0	1	2	3	0	1	2	3
a_1	0	1	2	3	0.05	0.10	0.25	0.60
a_2	0	1	—	—	0.10	0.90	—	—
a_3	0	1	—	—	0.10	0.90	—	—
a_4	0	1	—	—	0.10	0.90	—	—
a_5	0	1	2	—	0.10	0.30	0.60	—
a_6	0	1	2	—	0.05	0.25	0.70	—

Table 2: Boolean variables of arcs of Figure 3.

Variable	Capacity	Probability
$a_{1,0}$	0	0.05
$a_{1,1}$	1	0.10
$a_{1,2}$	2	0.25
$a_{1,3}$	3	0.60
$a_{2,0}$	0	0.10
$a_{2,1}$	1	0.90
$a_{3,0}$	0	0.10
$a_{3,1}$	1	0.90
$a_{4,0}$	0	0.10
$a_{4,1}$	1	0.90
$a_{5,0}$	0	0.10
$a_{5,1}$	1	0.30
$a_{5,2}$	2	0.60
$a_{6,0}$	0	0.05
$a_{6,1}$	1	0.25
$a_{6,2}$	2	0.70

4.3.1. Algorithm MFN_OBDD Process. For the example depicted in Figure 3, the steps of executing algorithm MFN_OBDD are as follows: (1) each state of an arc corresponds to a Boolean variable. For arc a_1 with four states, four variables, $a_{1,0}$, $a_{1,1}$, $a_{1,2}$, and $a_{1,3}$, are set up. The Boolean variables for Figure 3 are listed in Table 2, and the variable order is specified with $\Pi : a_{6,2} \succ a_{6,1} \succ a_{6,0} \succ a_{5,2} \succ \cdots \succ a_{1,3} \succ a_{1,2} \succ a_{1,1} \succ a_{1,0}$. (2) A flow, $\text{flow}(3)_1 = (2, 1, 0, 0, 2, 1)$, is obtained, and $\bar{\mathbf{c}} = (3, 1, 1, 1, 2, 2)$. (3) For each variable, $a_{i,j}$_BDD is constructed. The BDDs of the variables $a_{1,0}$–$a_{1,3}$ are shown in Figures 4(b)–4(e), respectively. (4) According to the variable order, a_i_OBDD of all arcs are generated applying the operation **Or**. Figures 4(f)–4(k) show the OBDDs of a_1–a_6. (5) f_1^d_OBDD shown in Figure 4(a) is obtained using the operation **And**; then d_OBDD = d_OBDD **Or** f_1^d_OBDD, and the OBDD is the same as Figure 5(a). (6) A new flow $\text{flow}(3)_2 = (2, 1, 1, 0, 1, 2)$ is obtained, and Steps 2–5 are repeated. Thus, f_2^d_OBDD is shown as Figure 5(b); the new d_OBDD is as depicted in Figure 6(a). (7) To obtain a new flow $\text{flow}(3)_3 = (3, 0, 1, 0, 2, 1)$ again, f_3^d_OBDD is as depicted in Figure 5(c), and the new d_OBDD is as depicted in Figure 6(b). (8) No more new flows are obtained. Thus, the OBDD in Figure 6(b) is the result d_OBDD, and we traverse d_OBDD with probabilities based on Lemma 3 to calculate the reliability with $d = 3$; $\text{Rel}(3) = 0.611415$.

4.3.2. Algorithm MFN_MDD Process. For the example depicted in Figure 3, steps in executing algorithm MFN_MDD are as follows: (1) each arc is a multistate variable, and the variable order is specified with $\Pi' : a_6 \succ a_5 \succ \cdots \succ a_2 \succ a_1$. (2) A flow, $\text{flow}(3)_1 = (2, 1, 0, 0, 2, 1)$, is obtained. (3) For each variable, a_i_MDD is constructed. In Figure 7, MDDs of a_1–a_6 are shown. (4) According to the variable order, f_1^d_MDD shown in Figure 8(a) is generated using operation **And**; then, d_MDD = d_MDD **Or** f_1^d_MDD, and the OBDD is the same as that depicted in Figure 8(a). (5) $\text{flow}(3)_2 = (2, 1, 1, 0, 1, 2)$ is obtained, and Steps 2–4 are repeated. Thus, f_2^d_MDD is shown as Figure 8(b), and the new d_MDD is shown as Figure 9(a). (6) $\text{flow}(3)_3 = (3, 0, 1, 0, 2, 1)$, and f_3^d_MDD is shown as Figure 8(c) and the new d_MDD as Figure 9(b). (7) The OBDD depicted in Figure 9(b) is the result d_MDD, and the reliability with $d = 3$ is calculated by traversing the d_OBDD with probabilities based on Lemma 4; $\text{Rel}(3) = 0.611415$.

4.4. Complexity Analysis of the Algorithms. For the algorithm MFN_OBDD, Step 1 takes n CPU time, and a flow(d) is obtained in Step 2 in $O(m^3)$ CPU time according to Corollary 1 in [1]; the operation **And** or **Or** of OBDD takes $O(n_1 n_2)$ time, where n_1 and n_2 denote the numbers of the two operated OBDDs, respectively. If $|C| = \max\{|\mathbf{C}_i|\}$ represents the maximum number of states for the arcs, then generating OBDD(a_i) takes $O(|C|(|C|-1)/2)$ time; and thus Steps 2.1–2.3 take $O(n \cdot (|C| \cdot (|C| - 1)/2) + 2^{|C|} \cdot (2^{n\cdot|C|} - 1)/(2^{|C|} - 1))$ time and looping Steps 2–4 take $O((m^3 + n \cdot |C|^2) \cdot \#T)$ time, where $\#T$ denotes the number of flow(d) that is unspecified, and the maximum flow is NP-hard, and $\#T$ in theory grows exponentially. According to Theorem 1 in [17], traversing d_OBDD to obtain the reliability takes $O(|C| \cdot (2^n - 1))$ time in the worst case. Therefore, the algorithm MFN_OBDD takes $O(|C| \cdot 2^n + (m^3 + n \cdot |C|^2) \cdot \#T)$ time.

For the proposed MFN_MDD algorithm, Steps 2.1-2.2 take $O(n \cdot |C| + n \cdot (n-1)/2)$ time; and looping of Steps 2–4 take $O((m^3 + n \cdot |C| + n(n-1)/2 + |C|^n) \cdot \#T)$ time. Theorem 1 in [17] states that it takes $O(|C|^n/n)$ to traverse the d_MDD in the worst case, so the computational complexity of MFN_MDD is $O((m^3 + n \cdot |C| + n^2 + |C|^n) \cdot \#T)$.

For the decomposition algorithm of [1], the ***while*** ... ***do*** loop repeats unspecified times $\#T_1$, and it takes another $\#T_2$

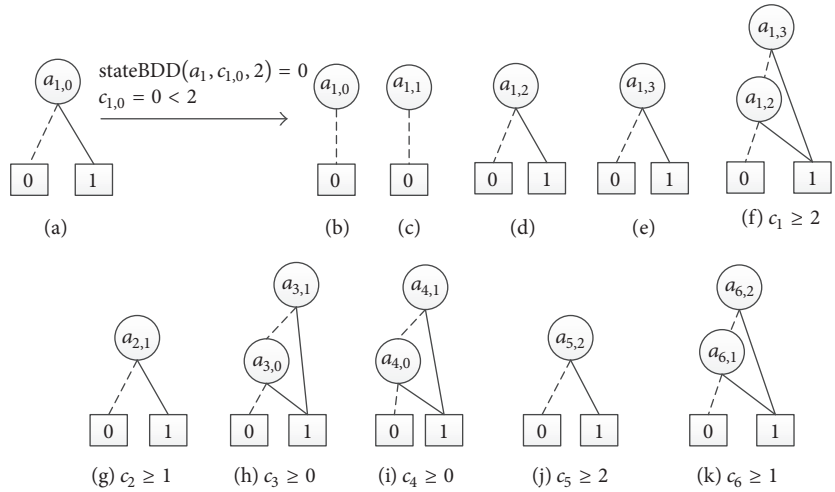

Figure 4: $a_{i,j}$_BDD and a_i_OBDD in MFN_OBDD: (a) the original BDD of $a_{1,0}$; (b)–(e) $a_{1,0}$_BDD–$a_{1,3}$_BDD; (f)–(k) a_1_OBDD–a_6_OBDD with $c_i \geq f^d_{1,i}$.

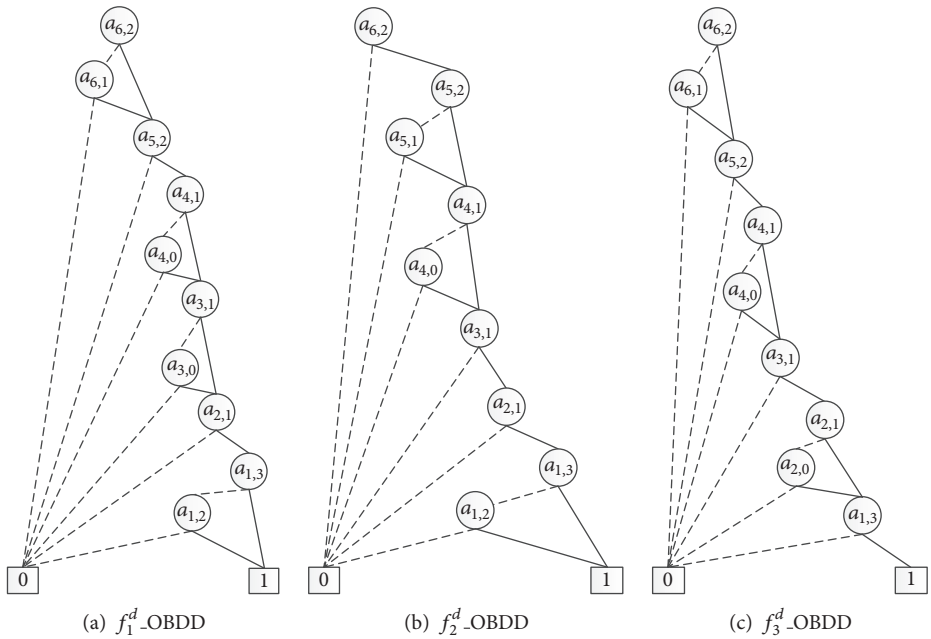

Figure 5: f^d_k_OBDD in MFN_OBDD: (a)–(c) f^d_1_OBDD()–f^d_3_OBDD.

time because the number of flow(d) is unspecified. Thus, the algorithm takes $O(m^3 \cdot \#T_1 \cdot \#T_2)$ at least. Obviously, the proposed algorithms have certain advantages.

The state vectors are stored in the decision diagrams (OBDDs or MDDs), which are tree date structures for the algorithm proposed in this paper. In the worst case, the OBDD takes $O(2^{n\cdot|C|})$ storage space in MFN_OBDD, and the MDD takes $O(|C|^n)$ storage space in MFN_MDD. And the decomposition algorithm of [1] requires $O(n \cdot \prod_{i=1}^{n} n_i)$ memory space in the worst case. However, because of the implicit expression in a decision diagram, the number of nodes is far less than the number in the worst case.

5. Experimental Results

In this paper, the algorithm MFN_OBDD is implemented based on the software package CUDD developed at the University of Colorado [27], while the algorithm MFN_MDD is programmed using the MDD package MEDDLY developed at Iowa State University. Both of the proposed algorithms are executed on a workstation running in Ubuntu 14.04 with 3.3 GHz, with 4 GB of RAM.

Figure 10 depicts a multistate network with 12 vertexes and 21 arcs from [1]. Vertex 1 is the source and vertex 12 is the sink. Each arc has three states with probabilities as specified

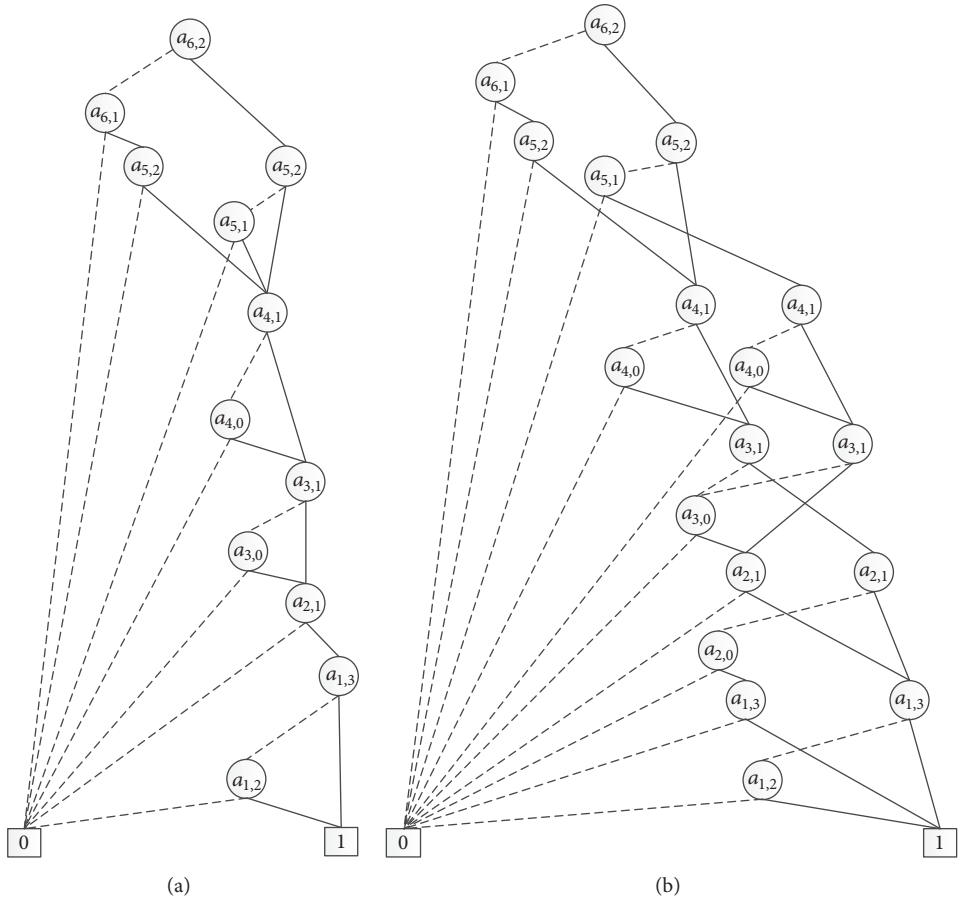

FIGURE 6: d_OBDDs: (a) d_OBDD after first loop; (b) result d_OBDD.

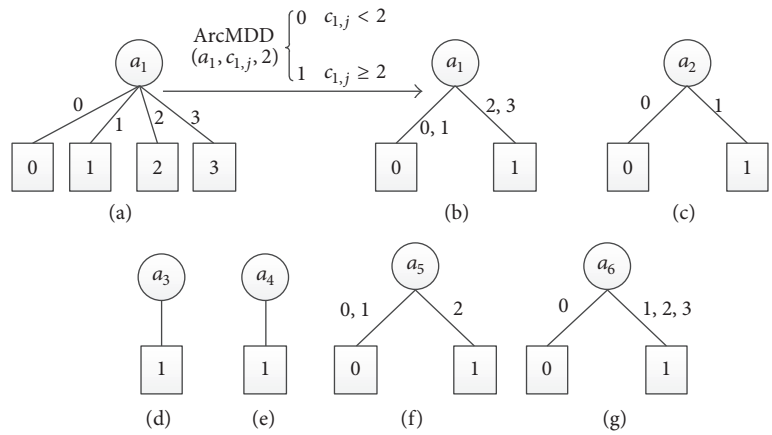

FIGURE 7: a_i_MDD in MFN_MDD: (a) the original MDD of a_1; (b)–(g) a_1_MDD–a_6_MDD, respectively.

in Table 3. Table 4 shows the experimental results for both the proposed algorithms and the decomposition algorithm of [1] with demand d ranging from 1 to 5. Studying Table 4 leads us to conclude the following:

(1) For the same demand, the number of decision diagram nodes generated in the MFN_MDD algorithm is less than the collections of [1] and MFN_OBDD's nodes is and no more than 65. When demand is small, for example, $d = 1, 2, 3$, the collections generated in the algorithm in [1] are less than MFN_OBDD's nodes, but more than MFN_OBDD's nodes when demand is large ($d = 4, 5$). In addition, the number of nodes in MFN_MDD is far less than the collections number

Table 3: States and probabilities of arcs for Figure 10.

Arcs	States			Probabilities		
	0	1	2	0	1	2
1	0	3	5	0.1163	0.0616	0.8221
2	0	3	5	0.1624	0.1224	0.7152
3	0	3	5	0.2014	0.900	0.7086
4	0	3	5	0.0689	0.1155	0.8156
5	0	3	5	0.1863	0.1366	0.6771
6	0	3	5	0.2244	0.0214	0.7542
7	0	3	5	0.1260	0.0495	0.8245
8	0	3	5	0.2220	0.1334	0.6445
9	0	3	5	0.1265	0.0762	0.7973
10	0	3	5	0.2993	0.0343	0.6664
11	0	3	5	0.3016	0.0813	0.6171
12	0	3	5	0.2385	0.0785	0.6830
13	0	3	5	0.3460	0.0269	0.6272
14	0	3	5	0.3512	0.0441	0.6048
15	0	3	5	0.0373	0.0830	0.8797
16	0	3	5	0.0326	0.0182	0.9492
17	0	3	5	0.0231	0.1268	0.8501
18	0	3	5	0.3935	0.0625	0.5440
19	0	3	5	0.0651	0.0457	0.8893
20	0	3	5	0.0052	0.0411	0.9537
21	0	3	5	0.0222	0.0192	0.9586

Table 4: Experimental results for different algorithms.

Demand	Reference [1]					Proposed algorithms					
						MFN_OBDD			MFN_MDD		Loop
	Collection[1]	Storage (bytes)	CPU time (s)	Loop	Node	Storage (bytes)	CPU time (s)	Node	Storage (bytes)	CPU time (s)	
1	294	51,744	0.016	636	611	7,332	0.024046	42	1,848	0.001928	17
2	294	51,744	0.016	636	611	7,332	0.216549	43	1,892	0.010049	153
3	294	51,744	0.016	636	611	7,332	1.3385	43	1,892	0.032208	969
4	9486	1,669,536	0.268	20,002	3,514	42,144	6.64096	51	2,244	0.475073	4,845
5	9486	1,669,536	0.268	20,002	3,514	42,144	27.7825	62	2,728	1.28560	20,349

1: collection is the main data structure of the algorithm in [1]. It contains the lower bound of edge state vector, the upper bound of edge state vector, the edge state flow vector, the vector of nonzero state edges, and two constants.

in [1]. However, according to the size of defined collection or node in different algorithms, the storage space is ordered as [1] > MFN_OBDD > MFN_MDD.

(2) The CPU time for executing the proposed algorithms increases with increasing d, and the CPU time for executing the MFN_MDD algorithm is less than that for the MFN_OBDD algorithm. The execution efficiency of MFN_MDD algorithm is higher than the algorithm in [1] with a small demand ($d = 1, 2$). And in other cases, the execution efficiency of proposed algorithm is slightly lower than the algorithm in [1].

(3) The loops number of proposed algorithms depend on the demand d. In most cases, it is much smaller than the loops number of the decomposition algorithm in [1].

In order to analyze the proposed algorithms, we select four more networks generated randomly; three of them, depicted in Figures 11(a)–11(c), are from [14], and that depicted in Figure 11(d) is from [1]. In this experiment, we specify that each arc of the five networks (including Figure 10) has three capacities, 0, 1, and 3, and the probabilities of states 0, 1, and 3 for each arc are 0.05, 0.25, and 0.70, respectively. Moreover, the demand is $d = 3$. Table 5 lists the experimental results for the proposed algorithms. Two conclusions can be drawn from studying Table 5:

(1) For the same network, the number of nodes and the number of variables of the DD generated in

Table 5: Experimental results for proposed algorithms.

Network			Reliability	MFN_OBDD			MFN_MDD		
Figure	Vertex	Arc		Variable	Node	CPU Time (s)	Variable	Node	CPU time (s)
Figure 11(a)	7	22	0.967046	43	1534	0.826896	16	50	0.046191
Figure 11(b)	10	21	0.970909	64	33933	9.37805	21	64	2.46028
Figure 11(c)	11	21	0.947015	64	8223	1.63739	21	64	0.304223
Figure 10	12	21	0.949288	64	6374	1.30824	21	42	0.01145
Figure 11(d)	15	30	0.974668	91	254307	340.043	30	91	253.156

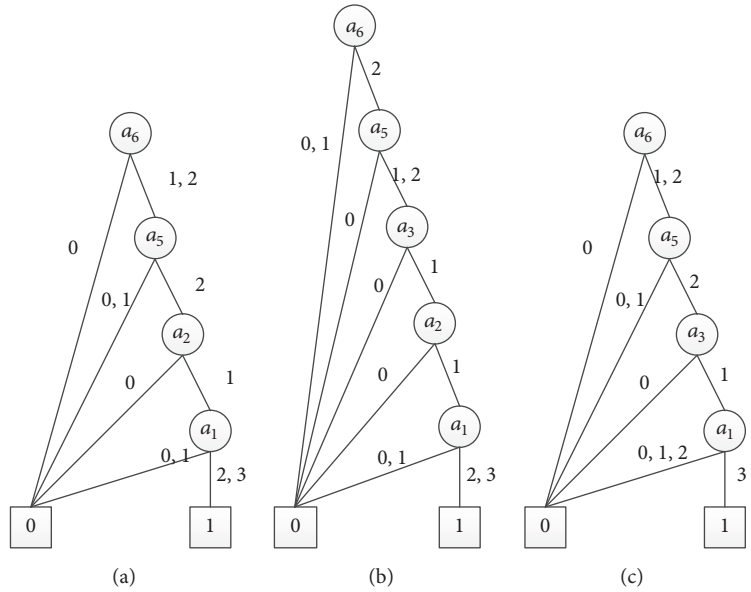

Figure 8: f_k^d_MDD in MFN_MDD: (a) f_1^d_MDD; (b) f_2^d_MDD; (c) f_3^d_MDD.

the MFN_MDD algorithm are both less than that of MFN_OBDD. The MFN_OBDD algorithm takes more CPU time to run. Because a state is a Boolean variable in the MFN_OBDD algorithm and the MFN_MDD algorithm makes an arc into a multivalue variable, the DD generated in the MFN_OBDD algorithm has a larger scale, thus leading to the conclusions described directly above.

(2) If the number of arcs is fixed and the number of vertexes is increased, such as in the networks depicted in Figures 11(b), 11(c), and 7, the experiments show that the number of nodes generated in either MFN_OBDD or MFN_MDD algorithm decreases. The CPU time also decreases. Therefore, in the case of the same arc number, both of the proposed algorithms have more advantages for sparse networks.

6. Conclusions

In this paper, we have extended the definition of a MFN model and have proposed the MFN_OBDD algorithm based on BDD and the MFN_MDD algorithm based on MDD to solve the problems in evaluating MFN reliability. The algorithms were applied to a simple MFN, and the attendant processes verified the feasibility of the algorithms. Experiments were carried out on five different networks, and the results show that (1) for the same network, the MFN_OBDD algorithm is more effective than the algorithm presented in [1] in terms of storage; in addition, due to the multibranch nature of a MDD, the storage efficiency of the MFN_MDD algorithm is better than that of the MFN_OBDD algorithm; (2) the runtime and the variables number of generated decision diagram in MFN_MDD algorithm are both less than those of the MFN_OBDD algorithm; (3) the storage space and CPU time of the proposed algorithms depend on the demand; and (4) the proposed algorithms have more advantages for sparse networks with the same number of arcs. The correctness, accuracy, and validity of the proposed algorithms were verified by the experiments. Furthermore, it would be worthwhile to improve the efficiency of algorithms' implementation.

Competing Interests

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

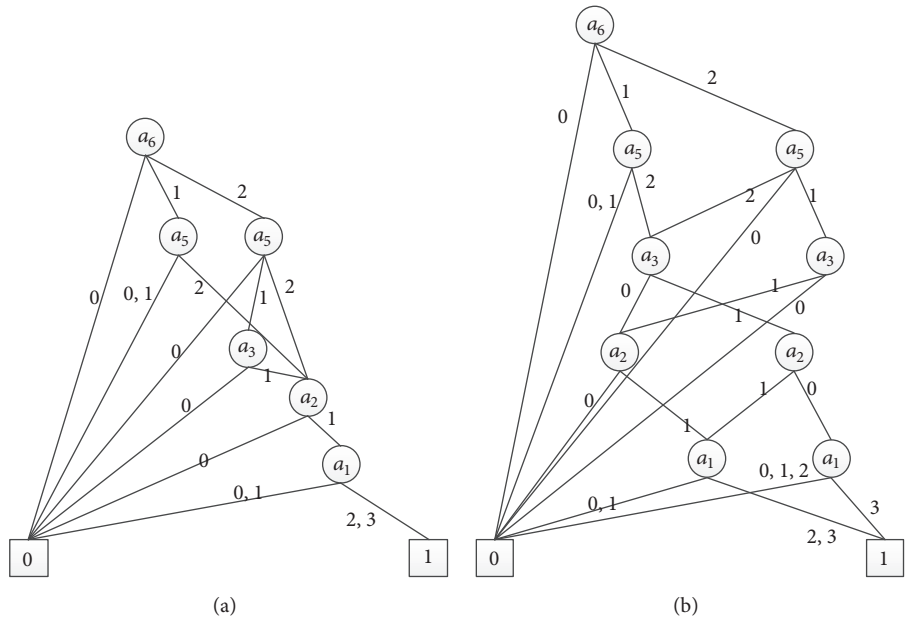

Figure 9: d_MDDs: (a) d_MDD after first loop; (b) result d_MDD.

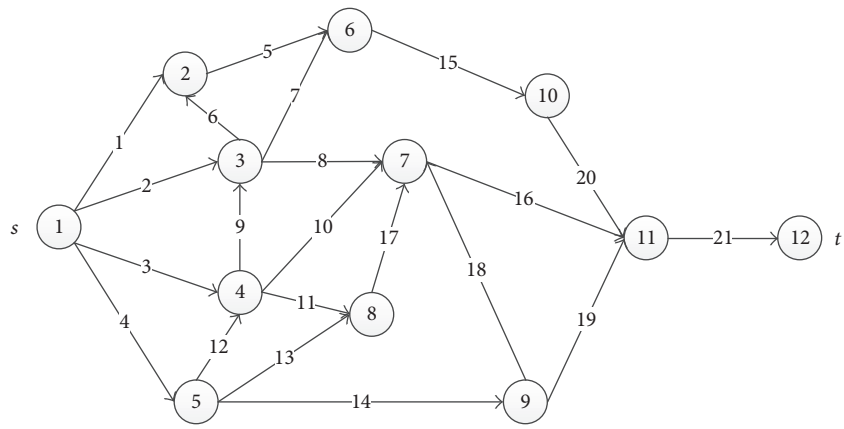

Figure 10: A multistate network.

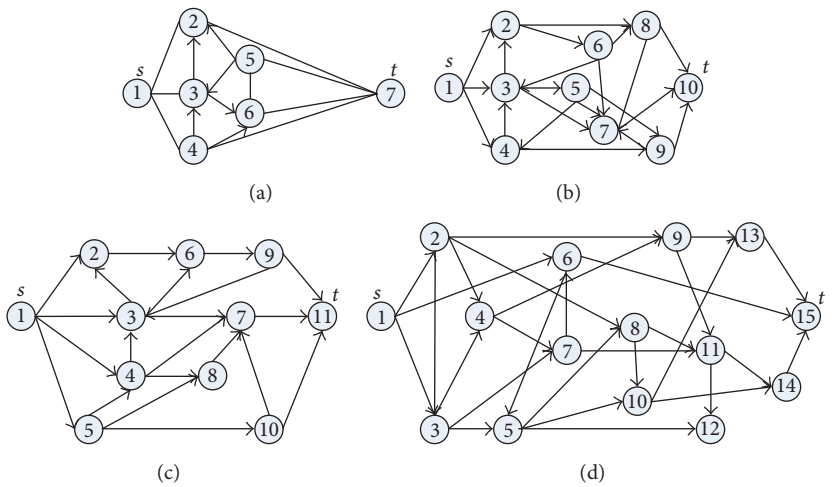

Figure 11: Four multistate networks.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 61363070), the Natural Science Foundation of Guangxi Province (Grant no. 2014GXNSFAA118354 and no. 2016GXNSFAA380054), the High Level Innovation Team of Guangxi Colleges and Universities and Outstanding Scholars Fund, and the Program for Innovative Research Team of Guilin University of Electronic Technology.

References

[1] 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.

[2] 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.

[3] W. C. Yeh, "A fast algorithm for quickest path reliability evaluations in multi-state flow networks," *IEEE Transactions on Reliability*, vol. 64, no. 4, pp. 1175–1184, 2015.

[4] R. Peng, H. Xiao, and H. Liu, *Reliability of Multi-State Systems with a Performance Sharing Group of Limited Size*, Reliability Engineering & System Safety, 2016.

[5] H. Yu, J. Yang, R. Peng, and Y. Zhao, "Reliability evaluation of linear multi-state consecutively-connected systems constrained by m consecutive and n total gaps," *Reliability Engineering & System Safety*, vol. 150, pp. 35–43, 2016.

[6] A. Lisnianski, "Application of extended universal generating function technique to dynamic reliability analysis of a multi-state system," in *Proceedings of the 2nd IEEE International Symposium on Stochastic Models in Reliability Engineering, Life Science and Operations Management (SMRLO '16)*, pp. 1–10, Beer Sheva, Israel, February 2016.

[7] C.-C. Jane and Y.-W. Laih, "Computing multi-state two-terminal reliability through critical arc states that interrupt demand," *IEEE Transactions on Reliability*, vol. 59, no. 2, pp. 338–345, 2010.

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

[9] C.-C. Jane, J.-S. Lin, and J. Yuan, "Reliability evaluation of a limited-flow network in terms of minimal cutsets," *IEEE Transactions on Reliability*, vol. 42, no. 3, pp. 354–361, 1993.

[10] Y.-K. Lin, "On a multicommodity stochastic-flow network with unreliable nodes subject to budget constraint," *European Journal of Operational Research*, vol. 176, no. 1, pp. 347–360, 2007.

[11] Y.-K. Lin, "Calculation of minimal capacity vectors through k minimal paths under budget and time constraints," *European Journal of Operational Research*, vol. 200, no. 1, pp. 160–169, 2010.

[12] Y.-K. Lin, "Time version of the shortest path problem in a stochastic-flow network," *Journal of Computational and Applied Mathematics*, vol. 228, no. 1, pp. 150–157, 2009.

[13] 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.

[14] W.-C. Yeh, C. Bae, and C.-L. Huang, "A new cut-based algorithm for the multi-state flow network reliability problem," *Reliability Engineering & System Safety*, vol. 136, pp. 1–7, 2015.

[15] W.-C. Yeh, "Evaluating the reliability of a novel deterioration-effect multi-state flow network," *Information Sciences*, vol. 243, no. 18, pp. 75–85, 2013.

[16] Y.-K. Lin, C.-T. Yeh, and P.-S. Huang, "A hybrid ant-tabu algorithm for solving a multistate flow network reliability maximization problem," *Applied Soft Computing*, vol. 13, no. 8, pp. 3529–3543, 2013.

[17] 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.

[18] Z. Yan, C. Nie, R. Dong, X. Gao, and J. Liu, "A novel OBDD-based reliability evaluation algorithm for wireless sensor networks on the multicast model," *Mathematical Problems in Engineering*, vol. 2015, Article ID 269781, 14 pages, 2015.

[19] X. Zang, D. Wang, H. Sun, and K. S. Trivedi, "A BDD-based algorithm for analysis of multistate systems with multistate components," *IEEE Transactions on Computers*, vol. 52, no. 12, pp. 1608–1618, 2003.

[20] A. Shrestha, L. Xing, and Y. Dai, "Decision diagram based methods and complexity analysis for multi-state systems," *IEEE Transactions on Reliability*, vol. 59, no. 1, pp. 145–161, 2010.

[21] Y. Mo, L. Xing, S. V. Amari, and J. Bechta Dugan, "Efficient analysis of multi-state k-out-of-n systems," *Reliability Engineering & System Safety*, vol. 133, pp. 95–105, 2015.

[22] A. Shrestha and D. W. Coit, "Multi-state component importance analysis using multi-state multi-valued decision diagrams," in *Proceedings of the International Conference on Reliability, Maintainability and Safety (ICRMS '09)*, pp. 99–103, Chengdu, China, July 2009.

[23] M. Kvassay, E. Zaitseva, J. Kostolny et al., "Reliability analysis of multi-state systems based on tools of multiple-valued logic," in *Proceedings of the IEEE International Conference on Computer as a Tool (EUROCON '15)*, Salamanca, Spain, 2015.

[24] R. Terruggia and A. Bobbio, "QoS analysis of weighted multi-state probabilistic networks via decision diagrams," in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2010)*, 54, pp. 2010–41, Vienna, Austria, September 2010.

[25] W. C. Yeh, "New method in searching for all minimal paths for the directed acyclic network reliability problem," *IEEE Transactions on Reliability*, vol. 65, no. 3, pp. 1263–1270, 2016.

[26] I. Tien and A. Der Kiureghian, "Algorithms for Bayesian network modeling and reliability assessment of infrastructure systems," *Reliability Engineering & System Safety*, vol. 156, pp. 134–147, 2016.

[27] F. S. Cudd, "Cu decision diagram package release 2.3.1," http://vlsi.colorado.edu/.