IEICE TRANS. INF. & SYST., VOL.E97–D, NO.9 SEPTEMBER 2014

PAPER Special Section on Multiple-Valued Logic and VLSI Computing

On Optimizations of Edge-Valued MDDs for Fast Analysis of Multi-State Systems*

Shinobu NAGAYAMA[†a)], **Tsutomu SASAO**[††b)], ***Members***, **Jon T. BUTLER**[†††c)], **Mitchell A. THORNTON**[††††d)], ***and*** **Theodore W. MANIKAS**[††††e)], ***Nonmembers***

SUMMARY In the optimization of decision diagrams, variable reordering approaches are often used to minimize the number of nodes. However, such approaches are less effective for analysis of multi-state systems given by monotone structure functions. Thus, in this paper, we propose algorithms to minimize the number of edges in an edge-valued multi-valued decision diagram (EVMDD) for fast analysis of multi-state systems. The proposed algorithms minimize the number of edges by grouping multi-valued variables into larger-valued variables. By grouping multi-valued variables, we can reduce the number of nodes as well. To show the effectiveness of the proposed algorithms, we compare the proposed algorithms with conventional optimization algorithms based on a variable reordering approach. Experimental results show that the proposed algorithms reduce the number of edges by up to 15% and the number of nodes by up to 47%, compared to the conventional ones. This results in a speed-up of the analysis of multi-state systems by about three times.

key words: *minimization algorithm of the number of edges, EVMDDs, grouping variables for optimization of decision diagrams, multi-state systems, system analysis using decision diagrams*

1. Introduction

Multi-state systems are widely used to model various fault tolerant systems including computer server systems, telecommunication systems, water, gas, electrical power distribution systems, flight control systems, and nuclear power plant monitoring systems [2], [3], [17], [21], [23]. In this system model, levels of performance, reliability, safety, efficiency, power consumption, etc. are represented as states.

To design dependable fault tolerant systems, intensive analysis of multi-state systems using various assessment measures for identifying critical components and system weaknesses is indispensable. Among them, assessing the probability of each state of a multi-state system is essential to the design of a dependable fault tolerant system [21], [23]. Thus, in this paper, we focus on computing the probabilities of system states. Since this is very time-consuming, many analysis methods have been proposed to shorten analysis time. Among them, methods based on binary decision diagrams (BDDs) [1], [2], [4], [23] and multi-valued decision diagrams (MDDs) [9], [15], [20], [21] have attracted much attention, since they hold promise as fast analysis methods.

In analysis methods based on decision diagrams (DDs), optimization of DDs is very important to reduce memory size and runtime for analysis. Most existing optimization algorithms for DDs use variable reordering approaches [5]–[7], [11], [12], [18]. However, for analysis of multi-state systems in which states of some components occur depending on states of other components [10], the order of some variables can be fixed. This is because conditional probabilities $P(B|A)$ are computed to analyze such systems, and $P(B|A)$ cannot be computed unless the value of A is decided prior to B. In addition, as we will show in Sect. 5, optimization over monotone structure functions is surprisingly unaffected by permuting variables. Thus, another approach that does not change the order of variables is more robust and effective for analysis of a wide range of systems.

In this paper, we use a variable grouping approach for optimization of DDs [13]. In many uses of DDs, minimization of the number of nodes is the objective of optimization. However, minimization of the number of nodes by grouping variables is trivial, and it is not always effective for fast analysis of multi-state systems. Thus, we propose algorithms to minimize *the number of edges* in an edge-valued multi-valued decision diagram (EVMDD) [14], [15] by grouping multi-valued variables into larger-valued variables. By grouping variables, we can reduce not only the number of edges, but also the number of nodes effectively, resulting in faster analysis of multi-state systems.

This paper is organized as follows: Section 2 defines multi-state systems, EVMDDs, and variable grouping. Section 3 introduces the analysis method of multi-state systems using MDDs and EVMDDs, and in Sect. 4, we propose algorithms to minimize the number of edges in an EVMDD. Experimental results are shown in Sect. 5.

Manuscript received November 28, 2013.
Manuscript revised April 10, 2014.

[†]The author is with the Department of Computer and Network Engineering, Hiroshima City University, Hiroshima-shi, 731–3194 Japan.
[††]The author is with the Department of Computer Science, Meiji University, Kawasaki-shi, 214–8571 Japan.
[†††]The author is with the Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA 93943–5121 USA.
[††††]The authors are with the Department of Computer Science and Engineering, Southern Methodist University, Dallas, Texas 75275–0122 USA.

*This paper is an extension of [16].

a) E-mail: s_naga@hiroshima-cu.ac.jp
b) E-mail: sasao@cs.meiji.ac.jp
c) E-mail: jon_butler@msn.com
d) E-mail: mitch@lyle.smu.edu
e) E-mail: manikas@lyle.smu.edu

DOI: 10.1587/transinf.2013LOP0011

2. Preliminaries

This section defines multi-state systems, structure functions, EVMDDs to represent structure functions, and variable grouping.

2.1 Multi-State Systems and Structure Functions

Definition 1: A **multi-state system** is a model of a system that represents, as a state, a capability, such as performance, capacity, or reliability. There are usually more than two states, and so a multiple-valued analysis is required. When components in a system are modeled as well, it is called a **multi-state system with multi-state components**. In this paper, it is simply called a multi-state system.

Definition 2: A state of a multi-state system depends only on states of components in the system. A system with n components can be considered as a multi-valued function $f(x_1, x_2, \ldots, x_n)$: $R_1 \times R_2 \times \ldots \times R_n \rightarrow M$, where each x_i represents a component with r_i states, $R_i = \{0, 1, \ldots, r_i - 1\}$ is a set of the states, and $M = \{0, 1, \ldots, m - 1\}$ is a set of the m system states. This multi-valued function is called a **structure function** of the multi-state system.

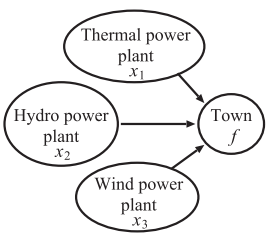
Definition 3: A structure function $f(x_1, x_2, \ldots, x_n)$ is **monotone increasing** iff, for all $\alpha, \beta \in R_i$, where $\alpha \leq \beta$,

$$f(x_1, x_2, \ldots, x_{i-1}, \alpha, x_{i+1}, \ldots, x_n) \leq f(x_1, x_2, \ldots, x_{i-1}, \beta, x_{i+1}, \ldots, x_n).$$

In many applications, states of a system and its components are totally ordered, and a deterioration of a component in the system causes a deterioration of the whole system. Thus, structure functions are usually monotone increasing when a value is assigned to each state in ascending order (i.e. the worst state is 0 and the best state is $m - 1$ or $r_i - 1$).

Example 1: Figure 1 (a) shows a multi-state system for an electrical power distribution system. In this figure, the thermal power plant x_1, the hydro power plant x_2, and the wind power plant x_3 have three states which correspond to supply levels: 0 (breakdown), 1 (partially supply), and 2 (fully supply). And, the system has six states which correspond to the percentage of area of a town that is blacked out: 0 (complete blackout), 1 (90% blackout), 2 (60% blackout), 3 (30% blackout), 4 (10% blackout), and 5 (0% blackout).

In this way, by assigning a value to each state in ascending order, we obtain the 6-valued monotone increasing structure function f shown in Fig. 1 (b). Note that Fig. 1 (b) shows a part of the $3^3 = 27$ entry table since it is too large to be included in its entirety. □



(a) Multi-state system.

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	0	2	1
0	1	0	1
0	1	1	2
	⋮		⋮
2	2	2	5

(b) Structure function.

Fig. 1 Multi-state system for an electrical power distribution system and its structure function.

2.2 Edge-Valued Multi-Valued Decision Diagrams

Definition 4: A **multi-valued decision diagram (MDD)** is a rooted directed acyclic graph representing a multi-valued function f. The MDD is obtained by repeatedly applying the Shannon expansion to the multi-valued function [8]. It consists of non-terminal nodes representing sub-functions obtained from f by assigning values to certain variables. It also has terminal nodes representing function values. Each non-terminal node has multiple outgoing edges that correspond to the values of a multi-valued variable. The MDD is ordered; i.e., the order of variables along any path from the root node to a terminal node is the same. In addition, the MDD is reduced; i.e., the following two reduction rules are applied to it:

1. Share equivalent sub-graphs.
2. Delete non-terminal nodes whose outgoing edges all point to the same node v, and redirect edges, that point to the deleted node, to v.

When an MDD represents a function for which multi-valued variables have different domains, it is a heterogeneous MDD [13]. In the following, the term 'MDD' refers to a heterogeneous MDD.

Definition 5: An **edge-valued MDD (EVMDD)** [14] is an extension of the MDD, and represents a multi-valued function. It consists of one terminal node representing 0 and non-terminal nodes with edges having integer weights; 0-edges always have zero weights. In an EVMDD, the function value is represented as a sum of weights for edges traversed from the root node to the terminal node.

EVMDDs are known as a compact representation for monotone increasing functions [14].

Example 2: Figure 2 and Fig. 3 show an ordinary MDD

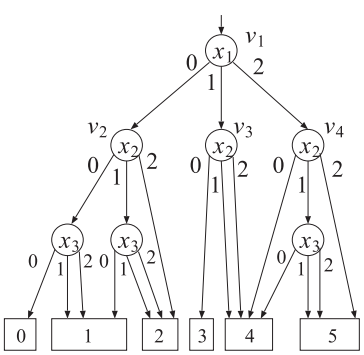


Fig. 2 MDD for the structure function.

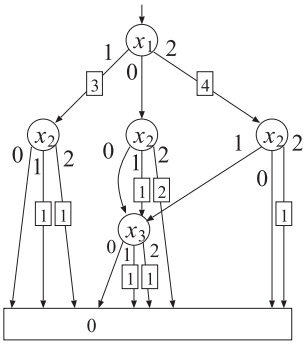


Fig. 3 EVMDD for the structure function.

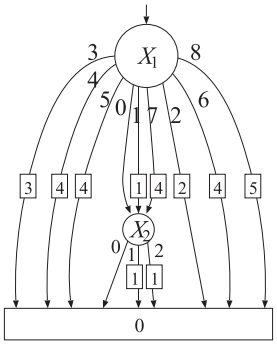


Fig. 4 EVMDD for the function $g(X_1, X_2)$.

and an EVMDD for the structure function of Example 1. For readability, some terminal nodes in the MDD are not combined. □

2.3 Variable Grouping

Definition 6: Let $X = (x_1, x_2, \ldots, x_n)$ be an ordered set of n multi-valued variables. Let

$$\begin{aligned} X_1 &= (x_1, x_2, \ldots, x_{k_1}), \\ X_2 &= (x_{k_1+1}, x_{k_1+2}, \ldots, x_{k_1+k_2}), \\ &\vdots \\ X_u &= (x_{k_1+k_2+\ldots+1}, x_{k_1+k_2+\ldots+2}, \ldots, x_n). \end{aligned}$$

Then, $(X_1, X_2, \ldots, X_u)$ is a **grouping** of X. Each ordered set $X_i = (x_{j+1}, x_{j+2}, \ldots, x_{j+k_i})$ forms a **super variable** whose domain is $\{0, 1, \ldots, r_{j+1} \times r_{j+2} \times \ldots \times r_{j+k_i} - 1\}$, where $|X_i| = k_i \geq 1$ and $k_1 + k_2 + \ldots + k_u = n$. Note that the order of the original multi-valued variables is preserved in a grouping.

By considering each super variable X_i as a larger-valued variable, the original multi-valued function $f(x_1, x_2, \ldots, x_n) : R_1 \times R_2 \times \ldots \times R_n \rightarrow M$ can be converted into its **larger-valued input function** $g(X_1, X_2, \ldots, X_u) : P_1 \times P_2 \times \ldots \times P_u \rightarrow M$, where $P_i = \{0, 1, \ldots, r_{j+1} \times r_{j+2} \times \ldots \times r_{j+k_i} - 1\}$.

Example 3: When the multi-valued variables x_1, x_2, x_3 in Example 1 are grouped into two super variables, we have

$$X_1 = (x_1, x_2) \quad \text{and} \quad X_2 = (x_3).$$

Note that since x_1 and x_2 are 3-valued variables, the super variable X_1 consisting of x_1 and x_2 is a 9-valued variable. The EVMDD representing the obtained function $g(X_1, X_2)$ is shown in Fig. 4. □

3. Analysis Methods Using MDDs and EVMDDs

Definition 7: The probability that a structure function f has the value σ is denoted by $P_s(f = \sigma)$, where $\sigma \in \{0, 1, \ldots, m-1\}$. The probability that a component x_i has the value γ is denoted by $P_c(x_i = \gamma)$, where $\gamma \in \{0, 1, \ldots, r_i - 1\}$.

An analysis of multi-state systems is to solve the following problem:

Problem 1: Given a structure function f of a multi-state system and the probability of each state of each component $P_c(x_i = \gamma)$, compute the probability of each state of the multi-state system $P_s(f = \sigma)$. For simplicity, we assume that the probabilities of all component states are independent of each other.

3.1 Analysis Method Using MDDs

Problem 1 can be solved using *node traversing probabilities* in an MDD that are introduced to compute the average path length on an MDD [12].

Definition 8: In an MDD, a sequence of edges and nodes leading from the root node to a terminal node is a **path**. The **node traversing probability**, denoted by $NTP(v_i)$, is the probability that an assignment of values to variables selects a path that includes the node v_i.

Since terminal nodes of an MDD for a structure function represent system states, node traversing probabilities of terminal nodes correspond to the probabilities of system states. The node traversing probabilities can be computed by visiting each node only once in the breadth first order from the root node. Thus, the time complexity of this analysis method is $O(N_M)$, where N_M is the number of nodes in an MDD [9], [20], [21].

Example 4: Let us compute node traversing probabilities for the MDD in Fig. 2. Assume that all states of each component occur with the same probability, 1/3.

First, we have $NTP(v_1) = 1$ for the root node v_1 since the root node occurs in all paths. Then, we compute $NTP(v_2) = NTP(v_1) \times 1/3$, $NTP(v_3) = NTP(v_1) \times 1/3$, and $NTP(v_4) = NTP(v_1) \times 1/3$ in a breadth first order. Similarly, by computing NTPs in a top-down manner, and by summing all NTPs received from parent nodes at re-convergence nodes, we have the node traversing probabilities of terminal nodes: $NTP(0) = 1/27$, $NTP(1) = 2/27 + 1/27 = 1/9$, $NTP(2) = 2/27 + 1/9 = 5/27$, $NTP(3) = 1/9$, $NTP(4) = 2/9 + 1/9 + 1/27 = 10/27$, and $NTP(5) = 2/27 + 1/9 = 5/27$. □

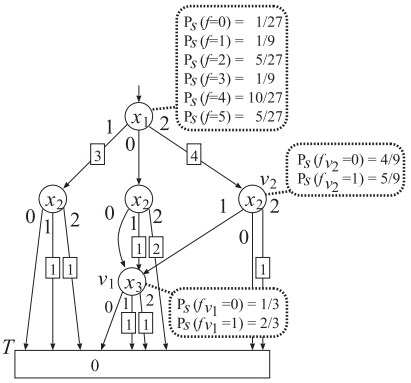


Fig. 5 Analysis of the multi-state system using EVMDD.

3.2 Analysis Method Using EVMDDs

To solve Problem 1 for large systems more efficiently, a method using EVMDDs has been proposed [15]. The method represents given structure functions using EVMDDs, and computes probabilities for a structure function by merging probabilities for sub-functions represented by nodes in a bottom-up manner.

Example 5: Let us compute the probability of each state of the multi-state system using the EVMDD in Fig. 5. Note that this corresponds to the system whose MDD is shown in Fig. 2. Assume that all states of each component occur with the same probability, 1/3.

First, we have $P_s(f_T = 0) = 1$ at the terminal node T. Then, we compute probabilities for a sub-function f_{v_1} at node v_1. Since this node has two edges pointing to T whose values are 1, and the two edges represent $f_{v_1} = 1$, we have

$$\begin{aligned} P_s(f_T = 0) \times P_c(x_3 = 1) &= 1/3, \\ P_s(f_T = 0) \times P_c(x_3 = 2) &= 1/3, \text{ and thus,} \\ P_s(f_{v_1} = 1) &= P_s(f_T = 0) \times P_c(x_3 = 1) \\ &\quad + P_s(f_T = 0) \times P_c(x_3 = 2) \\ &= 2/3. \end{aligned}$$

Thus, $P_s(f_{v_1} = 0) = 1/3$ and $P_s(f_{v_1} = 1) = 2/3$ for v_1. At v_2, the probabilities at the terminal node and v_1 are multiplied by 1/3, and they are merged in each function values of f_{v_2}. Thus, $P_s(f_{v_2} = 0) = 4/9$ and $P_s(f_{v_2} = 1) = 5/9$. Similarly, by performing the same computation at each node in a bottom-up manner, we have the following at the root node: $P_s(f = 0) = 1/27$, $P_s(f = 1) = 1/9$, $P_s(f = 2) = 5/27$, $P_s(f = 3) = 1/9$, $P_s(f = 4) = 10/27$, and $P_s(f = 5) = 5/27$. Note that these probabilities are identical to the node traversing probabilities at the terminal nodes in Example 4. □

The time complexity of this analysis method is $O(N_E)$, where N_E is the number of nodes in an EVMDD. Since in many applications, structure functions are monotone increasing, the functions are compactly represented by EVMDDs, and Problem 1 can be solved efficiently.

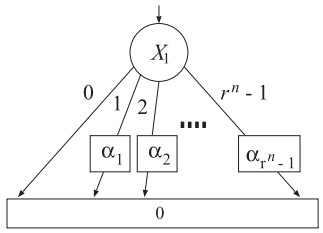


Fig. 6 EVMDD for a function $g(X_1)$, $X_1 = (x_1, x_2, \ldots, x_n)$.

3.3 Time Complexities of the Analysis Methods

The time complexities of the analysis methods using MDDs and EVMDDs are $O(N_M)$ and $O(N_E)$, as shown in the previous subsections. However, these are rough estimates. More precisely, the time complexities of the both methods can be expressed by the following form:

$$\sum_{i=0}^{N_\gamma} \left(\alpha_\gamma(i) + \sum_{j=0}^{R_\gamma(i)} \beta_\gamma(j) \right), \tag{1}$$

where $\alpha_\gamma(i)$ is the overhead for merging probabilities at each node, $\beta_\gamma(j)$ is the overhead for multiplying probabilities at each edge, N_γ is the number of nodes in an MDD or an EVMDD, $R_\gamma(i)$ is the number of edges of each node, and γ in this equation is stated to be either M or E for MDDs or EVMDDs. Let α_γ and β_γ be the maximum overheads, and R_γ be the maximum number of edges. Then, (1) can be approximated as follows:

$$N_\gamma(\alpha_\gamma + \beta_\gamma R_\gamma) = \alpha_\gamma N_\gamma + \beta_\gamma N_\gamma R_\gamma. \tag{2}$$

Thus, the computation time of the analysis methods can be shortened by reducing the number of nodes N_γ using algorithms based on variable reordering approaches [5]–[7], [11], [18].

We can minimize the number of nodes N_γ straightforwardly by grouping all n multi-valued variables of a given structure function into a super variable as shown in Fig. 6. In this case, although the number of nodes N_γ is only one, the number of edges R_γ in the node is r^n, where r is the domain size of each multi-valued variable. Since the time complexity is $O(r^n)$ in this example, minimization of the number of nodes by grouping variables does not always shorten computation time of the analysis methods.

In the optimization based on variable reordering, computation time can be shortened by minimization of the number of nodes N_γ since the number of edges R_γ in each node is constant. However, in the optimization based on variable grouping, minimization of N_γ can increase R_γ excessively. Since the $N_\gamma R_\gamma$ in (2) denotes the total number of edges in a decision diagram, minimization of the number of edges $N_\gamma R_\gamma$ is effective in the optimization based on variable grouping.

4. Minimization of the Number of Edges

Example 6: The EVMDD shown in Fig. 3 has 15 edges.

Algorithm 1: Minimization of the number of edges by grouping.

```
 1:  min_edge_grouping (EVMDD, the number of variables n) {
 2:    for(i = n; i ≥ 1; i = i - 1) {
 3:      min_edges = ∞ ;
 4:      for(k = 1; k ≤ limit[i]; k = k + 1) {
 5:        n_edges = nodes(EVMDD, i, k) × ∏_{j=0}^{k-1} r_{i+j} ;
 6:        n_edges = n_edges + lower_edges[i + k] ;
 7:        if (min_edges > n_edges) {
 8:          min_edges = n_edges ;
 9:          register the grouping k ;
10:        }
11:      }
12:      lower_table[i] = min_edges ;
13:    }
14:    return lower_table[1] ;
15:  }
```

On the other hand, the EVMDD shown in Fig. 4 has 12 edges, and it is the EVMDD with the minimum number of edges. If all the variables x_1, x_2, and x_3 are grouped into a single super variable as in Fig. 6, then an EVMDD obtained by this grouping has $3^3 = 27$ edges. □

As shown in Example 6, different groupings of variables produce EVMDDs with a different number of edges. Thus, there is an optimum grouping of variables that produces an EVMDD with the minimum number of edges. This section formulates a minimization problem of the number of edges in an EVMDD, and then presents minimization algorithms.

Problem 2: Given an EVMDD representing a structure function $f(x_1, x_2, \ldots, x_n)$, find a grouping of variables $(x_1, x_2, \ldots, x_n)$ that produces an EVMDD with the minimum number of edges.

Algorithm 1 shows pseudo-code to solve Problem 2. This algorithm is based on dynamic programing, and searches for the minimum number of edges for each sub-EVMDD sequentially from the bottom. In the following, for simplicity, we assume that the variable order for a given EVMDD is $x_1, x_2, \ldots, x_n$ from the top to the bottom.

Algorithm 1 is efficient because **limit[i]** prevents unnecessary iterations of the second for loop. This is shown by the following theorem.

Theorem 1: Let **nodes(EVMDD,** i**,** k**)** be the number of nodes in an EVMDD with respect to a super variable that consists of k variables from x_i to x_{i+k-1}, and let **edges(EVMDD,** i**)** be the number of edges associated with nodes in the given EVMDD representing variables from x_i to x_n. If, for some value of k, the following relation holds:

$$\textbf{nodes(EVMDD, } i\textbf{, } k\textbf{)} \times \prod_{j=0}^{k-1} r_{i+j} > \textbf{edges(EVMDD, } i\textbf{)},$$

then for any $k' \geq k$, the same relation holds:

$$\textbf{nodes(EVMDD, } i\textbf{, } k'\textbf{)} \times \prod_{j=0}^{k'-1} r_{i+j} > \textbf{edges(EVMDD, } i\textbf{)}.$$

Algorithm 2: Minimization of # of edges by grouping & ordering.

```
 1:  min_edge_g&o (EVMDD, the number of variables n) {
 2:    cost = min_edge_grouping(EVMDD, n) ;
 3:    do {
 4:      for(all multi-valued variables x_i) {
 5:        best_p = current position of x_i ;
 6:        for(all position p) {
 7:          Move x_i to position p ;
 8:          new_cost = min_edge_grouping(EVMDD, n) ;
 9:          if (new_cost < cost) {
10:            cost = n_edges ;
11:            best_p = p ;
12:            register the grouping ;
13:          }
14:        }
15:        Move x_i to best_p ;
16:      }
17:    } while (cost is reduced) ;
18:  }
```

(Proof) See Appendix.

This theorem states that, once the number of edges in an EVMDD obtained by grouping variables becomes larger than that in the original EVMDD, the number of edges cannot be reduced by grouping more variables. Thus, we can prune such redundant branching.

In the 5th line, **nodes(EVMDD, i, k)** denotes the number of root nodes for sub-EVMDDs from x_i to x_{i+k-1}. When k variables $x_i, x_{i+1}, \ldots, x_{i+k-1}$ are grouped into a super variable, each root node for the sub-EVMDDs corresponds to each node in an EVMDD with respect to the super variable, which has $\prod_{j=0}^{k-1} r_{i+j}$ edges. That is, the 5th line computes the number of edges in the EVMDD with respect to the super variable from x_i to x_{i+k-1}.

In the 6th line, the table **lower_edges[i + k]** stores the minimum number of edges computed for the lower-EVMDD from x_{i+k} to x_n. By summing this number and the number of edges computed in the 5th line, we have the number of edges in sub-EVMDDs from x_i to x_n.

The time complexity of Algorithm 1 is $O(n^2)$. However, the coefficient of n^2 is very small due to Theorem 1.

Since the proposed algorithm does not change the order of the original variables, it can be also applied to the analysis of multi-state systems in which states of some components occur depending on states of other components [10]. However, for the analysis of multi-state systems in which components are independent of each other, we can use both variable grouping and variable reordering approaches to reduce the number of edges furthermore. Algorithm 2 shows pseudo-code to minimize the number of edges using both Algorithm 1 and the sifting algorithm [5], [11], [18].

The sifting algorithm iteratively performs the following basic steps:

1. Change the current variable order.
2. Compute a cost.

Although the number of nodes is usually used as the cost, we use Algorithm 1 to compute the cost. To minimize the number of edges heuristically, Algorithm 2 computes the opti-

Table 1 Number of edges in MDDs and EVMDDs for m-state systems with n 3-state components.

n	m	CTT	w/o optimization		Ordering		Grouping				Ordering & Grouping	
			MDD	EVMDD	MDD	EVMDD	MDD	Ratio1	EVMDD	Ratio2	MDD	EVMDD
5	3	363	27	27	27	27	27	100%	27	100%	27	27
5	10	363	78	51	72	51	75	104%	48	94%	69	48
10	3	88,572	42	42	36	36	42	117%	42	117%	36	36
10	10	88,572	201	168	201	159	195	97%	162	102%	195	153
10	100	88,572	1,497	792	1,497	780	1,455	97%	750	96%	1,455	738
10	1,000	88,572	9,603	2,718	9,603	2,718	9,084	95%	2,364	87%	9,084	2,364
15	3	21,523,359	87	87	87	78	87	100%	87	112%	87	78
15	10	21,523,359	330	312	330	300	327	99%	309	103%	327	297
15	100	21,523,359	2,994	2,121	2,988	1,989	2,949	99%	2,076	104%	2,949	1,944
15	1,000	21,523,359	24,030	10,083	24,030	9,777	23,541	98%	9,597	98%	23,541	9,291
15	10,000	21,523,359	180,420	34,419	180,369	34,413	174,876	97%	31,212	91%	174,798	31,206
15	100,000	21,523,359	1,185,672	188,274	1,185,672	188,274	1,129,887	95%	159,768	85%	1,129,887	159,768

n: Number of 3-state components. m: Number of states for systems.
Ratio1: MDD with grouping / MDD with ordering × 100 (%). Ratio2: EVMDD with grouping / EVMDD with ordering × 100 (%).
The order of variables for MDDs and EVMDDs in "w/o optimization" is $x_1, x_2, \ldots, x_n$ (from top to bottom).

Table 2 Number of nodes in MDDs and EVMDDs for m-state systems with n 3-state components.

n	m	CTT	w/o optimization		Ordering		Grouping				Ordering & Grouping	
			MDD	EVMDD	MDD	EVMDD	MDD	Ratio1	EVMDD	Ratio2	MDD	EVMDD
5	3	364	12	10	12	10	10	83%	8	80%	10	8
5	10	364	36	18	34	18	33	97%	15	83%	31	15
10	3	88,573	17	15	15	13	15	100%	13	100%	13	11
10	10	88,573	77	57	77	54	67	87%	47	87%	67	44
10	100	88,573	599	265	599	261	505	84%	171	66%	505	167
10	1,000	88,573	4,201	907	4,201	907	3,300	79%	547	60%	3,300	547
15	3	21,523,360	32	30	32	27	30	94%	28	104%	30	25
15	10	21,523,360	120	105	120	101	117	98%	102	101%	117	98
15	100	21,523,360	1,098	708	1,096	664	1,003	92%	613	92%	1,003	569
15	1,000	21,523,360	9,010	3,362	9,010	3,260	8,119	90%	2,472	76%	8,119	2,370
15	10,000	21,523,360	70,140	11,474	70,123	11,472	61,732	88%	8,219	72%	61,706	8,217
15	100,000	21,523,360	495,224	62,759	495,224	62,759	417,581	84%	33,575	53%	417,581	33,575

Ratio1: MDD with grouping / MDD with ordering × 100 (%). Ratio2: EVMDD with grouping / EVMDD with ordering × 100 (%).

Table 3 Computation time (sec.) to optimize MDDs and EVMDDs for the systems.

n	m	Ordering		Grouping		Ordering & Grouping	
		MDD	EVMDD	MDD	EVMDD	MDD	EVMDD
5	3	* < 0.01	* < 0.01	* < 0.01	* < 0.01	* < 0.01	* < 0.01
5	10	* < 0.01	* < 0.01	* < 0.01	* < 0.01	* < 0.01	* < 0.01
10	3	* < 0.01	* < 0.01	* < 0.01	* < 0.01	* < 0.01	0.01
10	10	* < 0.01	0.01	* < 0.01	* < 0.01	* < 0.01	* < 0.01
10	100	* < 0.01	0.01	* < 0.01	* < 0.01	0.02	0.02
10	1,000	0.03	0.03	* < 0.01	* < 0.01	0.16	0.06
15	3	* < 0.01	* < 0.01	* < 0.01	* < 0.01	* < 0.01	* < 0.01
15	10	* < 0.01	* < 0.01	* < 0.01	* < 0.01	* < 0.01	0.01
15	100	0.04	0.04	* < 0.01	* < 0.01	0.04	0.07
15	1,000	0.16	0.22	* < 0.01	* < 0.01	0.60	0.54
15	10,000	3.97	1.35	0.05	0.01	31.63	4.16
15	100,000	19.97	3.04	1.13	0.06	385.76	17.42

* <: It was shorter than 1 msec., but could not be obtained precisely due to precision of the timer.

mum variable grouping while moving each variable x_i to all possible positions.

5. Experimental Results

To show the effectiveness of the proposed optimization algorithms for fast system analysis, we compare the proposed algorithms with the sifting algorithms for MDDs and EVMDDs [5], [11], [18]. In this experiment, we use monotone increasing structure functions randomly generated in [15] as benchmarks. This is because, unfortunately, benchmark structure functions of multi-state systems large enough to show the effectiveness of the proposed algorithms are unavailable. The randomly generated m-state systems with n 3-state components are analyzed using the optimized MDDs or EVMDDs, as shown in Sect. 3. The algorithms and the analysis methods are implemented on our own MDD package, and run on the following computer environment: CPU: Intel Core2 Quad Q6600 2.4GHz, memory: 4GB, OS: CentOS 5.7, and C-compiler: gcc -O3 (version 4.1.2).

Tables 1–4 show their experimental results. In these tables, the columns "w/o optimization," "Ordering," "Grouping," and "Ordering & Grouping" show the results obtained without any optimization, by the sifting algorithms, by Algorithm 1, and by Algorithm 2, respectively. And, for comparison, Tables 1 and 2 show the number of edges ($(3^{n+1} - 1)/2 - 1$) and the number of nodes ($(3^{n+1} - 1)/2$) in a complete ternary tree (CTT) that does not delete redundant nodes nor share equivalent sub-graphs. The tables show that MDDs and EVMDDs are several orders of magnitude

Table 4 Computation time (μsec.) to analyze m-state systems with n 3-state components.

n	m	w/o optimization		Ordering		Grouping				Ordering & Grouping	
		MDD	EVMDD	MDD	EVMDD	MDD	Ratio1	EVMDD	Ratio2	MDD	EVMDD
5	3	0.33	0.96	0.35	0.99	0.34	98%	0.98	98%	0.31	0.95
5	10	1.09	2.24	0.98	2.22	1.01	103%	2.06	92%	0.80	2.07
10	3	0.48	1.55	0.45	1.77	0.65	146%	1.56	88%	0.41	1.72
10	10	2.70	6.61	2.57	7.59	2.95	115%	6.39	84%	2.94	7.05
10	100	24.78	42.32	25.16	41.75	23.37	93%	33.29	80%	23.49	34.40
10	1,000	218.61	258.74	218.30	258.11	179.69	82%	155.89	60%	181.31	154.77
15	3	1.12	3.36	1.15	3.44	1.22	106%	3.11	91%	1.21	3.27
15	10	4.49	12.18	4.59	14.14	4.47	97%	12.48	88%	4.65	14.04
15	100	57.76	95.90	56.08	92.42	53.05	95%	86.76	94%	52.71	83.88
15	1,000	532.28	652.68	532.67	650.26	507.46	95%	500.55	77%	510.78	505.46
15	10,000	4,273.00	3,953.00	4,532.00	3,957.00	3,737.00	82%	2,397.00	61%	3,767.00	2,409.00
15	100,000	61,211.00	55,097.00	61,277.00	56,456.00	41,194.00	67%	18,257.00	32%	41,198.00	18,305.00

Ratio1: MDD with grouping / MDD with ordering × 100 (%). Ratio2: EVMDD with grouping / EVMDD with ordering × 100 (%).
The computation time is an average time obtained by running the same computation 1,000,000 times, and dividing its total time by 1,000,000.

smaller than CTTs.

From these tables, we can see that the sifting algorithms are less effective on the randomly generated monotone increasing systems since they reduce neither the number of edges nor the number of nodes very much. The difference between computation times of "w/o optimization" and "Ordering" in Table 4 is just within the margin of measurement error. On the other hand, Algorithm 1 based on variable grouping shortens the computation time for analyzing the multi-state systems significantly, especially for large systems. This is because Algorithm 1 reduces both the number of edges and the number of nodes.

Surprisingly, the computation time of analysis is reduced more than the number of edges and nodes are reduced, when m is large. Especially in analysis using EVMDDs, the computation time is significantly reduced. This is because a reduction in the number of nodes and edges reduces the overheads $\alpha_\gamma(i)$ and $\beta_\gamma(j)$ in (1). In the analysis method using EVMDDs, probabilities of function values at each node are merged at its parent node, as shown in Fig. 5. Thus, the overhead $\alpha_\gamma(i)$ increases as the number of function values at child nodes increases. Our optimization algorithm usually groups nodes near the root node into one node, as shown in Fig. 4. Since nodes near the root node tend to have many function values (i.e., large $\alpha_\gamma(i)$), this grouping yields a significant reduction in the computation time of analysis using EVMDDs.

Algorithm 2 using both variable reordering and variable grouping reduces the number of edges even more, but does not improve the speed of analysis very much. Although Algorithm 2 requires more time to optimize MDDs or EVMDDs than Algorithm 1, its improvement is small. On the other hand, Algorithm 1 is the fastest at optimizing MDDs or EVMDDs among the three algorithms, and its effect to shorten analysis time is large.

From these results, we can say that the proposed optimization algorithms are very effective for fast system analysis, since minimization of the number of edges by variable grouping reduces the number of nodes, as well as overhead for merging probabilities. Particularly, the proposed algorithms are more effective for the analysis method using EVMDDs since the overhead is significantly reduced.

6. Conclusion and Comments

This paper proposes minimization algorithms of the number of edges in an EVMDD for fast analysis of multi-state systems. The proposed algorithms minimize the number of edges by grouping multi-valued variables into larger-valued variables. By grouping multi-valued variables, we can also reduce the number of nodes and the overhead for merging probabilities. Experimental results show that the proposed algorithms reduce the number of edges by up to 15% and reduces the number of nodes by up to 47%, resulting in much faster analysis of multi-state systems. Since the algorithm based only on the variable grouping does not change the order of the original variables, it can also be applied to the analysis of multi-state systems in which states of some components occur depending on states of other components. Therefore, it is robust and effective for analysis of a wide range of systems. In addition, it can optimize EVMDDs quickly.

Acknowledgments

This research is partly supported by the JSPS Grant-in-Aid for Scientific Research (C), (No. 25330071), 2014. The reviewers' comments were helpful in improving this paper.

References

[1] J.D. Andrews and S.J. Dunnett, "Event-tree analysis using binary decision diagrams," IEEE Trans. Reliab., vol.49, no.2, pp.230–238, June 2000.
[2] Y.-R. Chang, S.V. Amari, and S.-Y. Kuo, "Reliability evaluation of multi-state systems subject to imperfect coverage using OBDD," Proc. 2002 Pacific Rim International Symposium on Dependable Computing (PRDC'02), pp.193–200, 2002.
[3] S.A. Doyle, J.B. Dugan, and F.A. Patterson-Hine, "A combinatorial approach to modeling imperfect coverage," IEEE Trans. Reliab., vol.44, no.1, pp.87–94, March 1995.
[4] S.A. Doyle and J.B. Dugan, "Dependability assessment using binary decision diagrams (BDDs)," 25th International Symposium on Fault-Tolerant Computing (FTCS), pp.249–258, June 1995.

[5] R. Drechsler, W. Günther, and F. Somenzi, "Using lower bounds during dynamic BDD minimization," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.20, no.1, pp.51–57, Jan. 2001.
[6] M. Fujita, Y. Matsunaga, and T. Kakuda, "On variable ordering of binary decision diagrams for the application of multi-level logic synthesis," EDAC, pp.50–54, March 1991.
[7] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of binary decision diagrams based on exchanges of variables," International Conference on Computer-Aided Design (ICCAD'91), pp.472–475, Nov. 1991.
[8] T. Kam, T. Villa, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, "Multi-valued decision diagrams: Theory and applications," Multiple-Valued Logic: An International Journal, vol.4, no.1-2, pp.9–62, 1998.
[9] T.W. Manikas, M.A. Thornton, and D.Y. Feinstein, "Using multiple-valued logic decision diagrams to model system threat probabilities," 41st International Symposium on Multiple-Valued Logic, pp.263–267, May 2011.
[10] T.W. Manikas, D.Y. Feinstein, and M.A. Thornton, "Modeling medical system threats with conditional probabilities using multiple-valued logic decision diagrams," 42nd International Symposium on Multiple-Valued Logic, pp.244–249, May 2012.
[11] D.M. Miller and R. Drechsler, "Augmented sifting of multiple-valued decision diagrams," 33rd International Symposium on Multiple-Valued Logic, pp.375–382, Tokyo, Japan, May 2003.
[12] S. Nagayama, A. Mishchenko, T. Sasao, and J.T. Butler, "Exact and heuristic minimization of the average path length in decision diagrams," J. Multiple-Valued Logic and Soft Computing, vol.11, no.5-6, pp.437–465, Aug. 2005.
[13] S. Nagayama and T. Sasao, "On the optimization of heterogeneous MDDs," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.24, no.11, pp.1645–1659, Nov. 2005.
[14] S. Nagayama, T. Sasao, and J.T. Butler, "A systematic design method for two-variable numeric function generators using multiple-valued decision diagrams," IEICE Trans. Inf. & Syst., vol.E93-D, no.8, pp.2059–2067, Aug. 2010.
[15] S. Nagayama, T. Sasao, and J.T. Butler, "Analysis of multi-state systems with multi-state components using EVMDDs," 42nd International Symposium on Multiple-Valued Logic, pp.122–127, May 2012.
[16] S. Nagayama, T. Sasao, and J.T. Butler, "Minimization of the number of edges in an EVMDD by variable grouping for fast analysis of multi-state systems," 43rd International Symposium on Multiple-Valued Logic, pp.284–289, May 2013.
[17] J.E. Ramirez-Marquez and D.W. Coit, "Composite importance measures for multi-state systems with multi-state components," IEEE Trans. Reliab., vol.54, no.3, pp.517–529, Sept. 2005.
[18] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," International Conference on Computer-Aided Design (ICCAD'93), pp.42–47, Nov. 1993.
[19] T. Sasao and M. Fujita (eds.), Representations of Discrete Functions, Kluwer Academic Publishers, 1996.
[20] L. Xing and J.B. Dugan, "Dependability analysis using multiple-valued decision diagrams," Proc. 6th International Conference on Probabilistic Safety Assessment and Management, June 2002.
[21] L. Xing and Y. Dai, "A new decision-diagram-based method for efficient analysis on multistate systems," IEEE Trans. Dependable and Secure Computing, vol.6, no.3, pp.161–174, 2009.
[22] S.N. Yanushkevich, D.M. Miller, V.P. Shmerko, and R.S. Stankovic, Decision Diagram Techniques for Micro- and Nanoelectronic Design, CRC Press, Taylor & Francis Group, 2006.
[23] X. Zang, D. Wang, H. Sun, and K.S. Trivedi, "A BDD-based algorithm for analysis of multistate systems with multistate components," IEEE Trans. Comput., vol.52, no.12, pp.1608–1618, Dec. 2003.

Appendix: Proof for Theorem 1 [16]

Suppose that for a value of k, the following relation holds:

$$\textbf{nodes(EVMDD, } i\textbf{, } k) \times \prod_{j=0}^{k-1} r_{i+j} > \textbf{edges(EVMDD, } i) \quad (\text{A}\cdot 1)$$

Then, we will prove that, for $k+1$, (A· 1) also holds.

By multiplying both sides of (A· 1) by r_{i+k}, we have

$$\textbf{nodes(EVMDD, } i\textbf{, } k) \times \prod_{j=0}^{k-1} r_{i+j} \times r_{i+k} > \textbf{edges(EVMDD, } i) \times r_{i+k}, \quad (\text{A}\cdot 2)$$

where r_{i+k} is the number of values of x_{i+k}.

From the definition of a super variable, the number of edges in an EVMDD with respect to a super variable that consists of $k+1$ variables from x_i to x_{i+k} is

$$\textbf{nodes(EVMDD, } i\textbf{, } k+1) \times \prod_{j=0}^{k} r_{i+j}.$$

Since **nodes(EVMDD, i, k)** is monotone increasing with respect to k, we have

$$\textbf{nodes(EVMDD, } i\textbf{, } k+1) \geq \textbf{nodes(EVMDD, } i\textbf{, } k)$$

and thus,

$$\textbf{nodes(EVMDD, } i\textbf{, } k+1) \times \prod_{j=0}^{k} r_{i+j} \geq \textbf{nodes(EVMDD, } i\textbf{, } k) \times \prod_{j=0}^{k} r_{i+j}. \quad (\text{A}\cdot 3)$$

From (A· 1), (A· 2), and (A· 3), the relation (A· 1) holds for $k+1$. Therefore, for any $k' \geq k$, the theorem holds. ∎

Shinobu Nagayama received the B.S. and M.E. degrees from the Meiji University, Kanagawa, Japan, in 2000 and 2002, respectively, and the Ph.D. degree in computer science from the Kyushu Institute of Technology, Japan, in 2004. He is now an Associate Professor at Hiroshima City University, Japan. He received the Outstanding Contribution Paper Awards from the IEEE Computer Society Technical Committee on Multiple-Valued Logic (MVL-TC) in 2005 and 2013 for papers presented at the International Symposium on Multiple-Valued Logic in 2004 and 2012, respectively, the Young Author Award from the IEEE Computer Society Japan Chapter in 2009, and the Outstanding Paper Award from the Information Processing Society of Japan (IPS) in 2010 for a paper presented at the IPSJ Transactions on System LSI Design Methodology. His research interest includes decision diagrams, analysis of multi-state systems, logic design for numeric function generators, regular expression matching, and multiple-valued logic.

Tsutomu Sasao received the BE, ME, and PhD degrees in electronics engineering from Osaka University, Osaka, Japan, in 1972, 1974, and 1977, respectively. He has held faculty/research positions at Osaka University, Japan, the IBM T.J. Watson Research Center, Yorktown Heights, New York, and the Naval Postgraduate School, Monterey, California. He is now a Professor of the Department of Computer Science at the Meiji University, Kawasaki, Japan. His research areas include logic design and switching theory, representations of logic functions, and multiple-valued logic. He has published more than nine books on logic design, including Logic Synthesis and Optimization, Representation of Discrete Functions, Switching Theory for Logic Synthesis, and Logic Synthesis and Verification, Kluwer Academic Publishers, 1993, 1996, 1999, and 2001, respectively. He has served as Program Chairman for the IEEE International Symposium on Multiple-Valued Logic (ISMVL) many times. Also, he was the Symposium Chairman of the 28th ISMVL held in Fukuoka, Japan, in 1998. He received the NIWA Memorial Award in 1979, Distinctive Contribution Awards from the IEEE Computer Society MVL-TC for papers presented at ISMVLs in 1986, 1996, 2003 and 2004, and Takeda Techno-Entrepreneurship Award in 2001. He has served as an Associate Editor of the IEEE Transactions on Computers. He is a fellow of the IEEE.

Jon T. Butler received the BEE and MEngr degrees from Rensselaer Polytechnic Institute, Troy, New York, in 1966 and 1967, respectively. He received the PhD degree from The Ohio State University, Columbus, in 1973. Since 1987, he has been a professor at the Naval Postgraduate School, Monterey, California. In 2010, he was promoted to Distinguished Professor, and in 2014, he became a Distinguished Professor Emeritus. From 1974 to 1987, he was at Northwestern University, Evanston, Illinois. During that time, he served two periods of leave at the Naval Postgraduate School, first as a National Research Council Senior Postdoctoral Associate (1980–1981) and second as the NAVALEX Chair Professor (1985–1987). He served one period of leave as a foreign visiting professor at the Kyushu Institute of Technology, Iizuka, Japan. His research interests include logic optimization numeric function generators, Boolean functions for cryptography, multi-state systems, multiple-valued logic, and reconfigurable computing. He has served on the editorial boards of the IEEE Transactions on Computers, Computer, and IEEE Computer Society Press. He has served as the editor-in-chief of Computer and IEEE Computer Society Press. He received the Award of Excellence, the Outstanding Contributed Paper Award, and a Distinctive Contributed Paper Award for papers presented at the International Symposium on Multiple-Valued Logic. He is a life fellow of the IEEE.

Mitchell A. Thornton received the B.S. in electrical engineering from Oklahoma State University, the M.S. in electrical engineering from the University of Texas at Arlington, the M.S. in computer science from Southern Methodist University, and the Ph.D. in computer engineering from Southern Methodist University in Dallas, Texas. He was employed in industry full-time at E-Systems, Inc from 1986 to 1991 and left there as a Senior Electronics Systems Engineer and at the Cyrix Corporation from 1991 to 1993 as a Design Engineer. He was a faculty member at the University of Arkansas and Mississippi State University and is now a Professor in the departments of electrical engineering and computer science and engineering at Southern Methodist University. He was designated as the J. Lindsey Embrey Chair in Computer Science and Engineering in 2004 and as a Gerald Ford Research Fellow at SMU in 2005. At SMU, he also is the Co-Director of the High Assurance Computing and Networking Research Laboratories (HACNet) at SMU. He has published more than 200 technical articles, 4 books, and is a named inventor on 2 US patents and 3 patents pending and he has consulted with and performed research for a variety of government and industrial organizations. He served as chair of the IEEE Technical Committee on Multiple-Valued Logic, chair of the IEEE-USA Licensure and Registration Committee, and chair of the NCEES Examination Development Committee for electrical engineering professional licensure. He is a licensed professional engineer in the states of Texas, Arkansas, and Mississippi. His research interests include physical computer security, electronic design automation, disaster and fault tolerance, and emerging technology. He is a senior member of the IEEE and ACM and is the editor of the digital circuits and systems series for Morgan & Claypool Publishers.

Theodore W. Manikas received the BSEE degree from Michigan State University, East Lansing, Michigan, the MSEE degree from Washington University, Saint Louis, Missouri, and the PhD degree from the University of Pittsburgh, Pennsylvania. He has held a faculty position at the University of Tulsa, Oklahoma, and has been a faculty member in the Department of Computer Science and Engineering at Southern Methodist University, Dallas, Texas, USA, since 2009. His focus areas include computer architecture, logic optimization, and system reliability and security. He is a member of the ACM, ASEE, and IEEE, and is a licensed Professional Engineer in the states of Texas and Oklahoma.