Enumeration of Minimal Cut Vectors for Performing Reliability Analysis Based on Multi-Valued Decision Diagrams

Miroslav Kvassay, Jozef Kostolny

***Abstract*—**A current issue in reliability engineering is analysis of complex systems that are composed of many heterogeneous components. Investigation of such systems requires development of new approaches that allow representing their structure in an efficient way. A prospective way to solve this issue is application of decision diagrams. However, their application requires development of new methods that allow performing reliability analysis by computing basic characteristics, such as system availability or unavailability, by studying importance of system components, or by enumerating circumstances under which an improvement of any system component that the system is composed of results in improvement of performance of the system. These circumstances are described by so-called minimal cut vectors. In this paper we present a simple algorithm for their enumeration assuming that the structure of a system is represented by a multi-valued decision diagram.

***Keywords*—**Minimal cut vector, multi-valued decision diagram, reliability, structure function.

I. Introduction

Development of highly reliable systems is a key task in many engineering fields. Such a development requires methods that are able to analyze current complex systems in an efficient way [1]. The key input to these methods is a mathematical model of the system, which has to carry all information that are necessary for answering key questions of reliability analysis, such as, how reliable is the system [2], [3], which components are most important for operation of the system [4], how to optimize system reliability [4], [5], or how to plan system maintenance [6]. A typical part of the model of a system is mathematical representation of its structure. This representation is known as structure function [2], [3].

The structure function carries information about dependency of performance of the system based on the performance of its components. The performance of the system and its components can be represented by integer numbers denoted as states. If only two states are defined for the system and all its components, which are state 0 (failure) and state 1 (functioning), then the system is known as a Binary-State System (BSS). Structure function of this kind of a system has the following form [2]:

$$\phi(x_1, x_2, \dots, x_n) = \phi(\boldsymbol{x}):\ \{0,1\}^n \rightarrow \{0,1\}, \tag{1}$$

where n denotes the number of the system components, x_i is a variable defining state of component i of the system, for $i = 1,2,\dots,n$, and $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ is a vector of components states (state vector). As one can see, this function formally agrees with a Boolean function. This allows us to use approaches and mathematical methodologies of Boolean logic in reliability analysis of BSSs [2], [7].

Models based on structure function (1) are useful in the analysis of systems in which any deviation from perfect functioning can result in a disaster, e.g., nuclear power plants [8] or aviation systems [9], in investigation of systems that are binary-state from their nature, e.g., logic circuits [10], or in studying consequences of system failure [11]. However, they are not

M. Kvassay, Faculty of Management Science and Informatics, University of Zilina, Zilina, Slovakia (e-mail: miroslav.kvassay@fri.uniza.sk).
J. Kostolny, Faculty of Management Science and Informatics, University of Zilina, Zilina, Slovakia (e-mail: jozef.kostolny@fri.uniza.sk).

very suitable for the analysis of systems that can operate at several performance levels, e.g., different kinds of distribution networks [1], [12], or systems composed of various types of components that are very different in their nature, e.g., socio-technical systems [1], such as healthcare systems [13], [14], or software systems, such as temporal databases studied in [15]. These and other systems which can be in one of more than two states are known as Multi-State Systems (MSSs). In this case, the structure function has the following form [16], [17]:

$$\begin{gathered}\phi(x_1, x_2, \dots, x_n) = \phi(\boldsymbol{x}): \\ \{0,1,\dots,m_1-1\} \times \{0,1,\dots,m_2-1\} \times \dots \times \{0,1,\dots,m_n-1\} \to \{0,1,\dots,m-1\},\end{gathered} \tag{2}$$

where, as in the case of a BSS, n agrees with the number of system components, x_i is a variable defining state of the i-th system component, for $i = 1,2,\dots,n$, $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ is a state vector defining states of all the system components, m denotes the number of system states/performance levels (value 0 agrees with complete failure, while value $m-1$ corresponds to perfect functioning of the system), and m_i agrees with the number of states of component i (similarly, state 0 means that the component fails, while state $m_i - 1$ agrees with its perfect functioning). Specially, if $m_1 = m_2 = \cdots = m_n = m$, then the MSS is homogeneous [3], [18]. Moreover, if $m_1 = m_2 = \cdots = m_n = m = 2$, structure function (2) agrees with structure function of a BSS, therefore, a BSS can be viewed as a special type of a homogeneous MSS.

Other parts of the model of a system under consideration are state probabilities of the system components denoted for MSSs as follows [19]:

$$\begin{gathered}p_{i,s} = \Pr\{x_i = s\}, \text{ for } i = 1,2,\dots,n,\ s = 0,1,\dots,m_i - 1, \\ \sum_{s=0}^{m_i-1} p_{i,s} = 1, \text{for } i = 1,2,\dots,n.\end{gathered} \tag{3}$$

In case of BSSs, $p_{i,0}$ is denoted as q_i and is known as unavailability of the i-th system component and $p_{i,1}$, known as component availability, is denoted simply as p_i [2], [11]. These state probabilities are very important for quantitative analysis because if we combine them with the knowledge of the structure function, then they can be used to compute global reliability measures, such as system availability or unavailability [2], [3], or to analyze importance of the system components for operation of the system [4].

Depending on the properties of structure function, MSSs as well as BSSs can be classified into two groups – coherent and noncoherent [4], [20]. A system is coherent if (a) all its components are relevant, i.e., for each component there exists a situation in which a change of component state results in a change of state of the system, and (b) the structure function is non-decreasing in all its arguments, i.e., there are no circumstances under which a failure/ degradation of any system component results in a repair/improvement of the system. If these assumptions are not satisfied, then the system is noncoherent. In the rest of the paper, we will assume that the system under consideration is coherent.

Structure function of a MSS is an integer function. Such a function can be represented in various ways [21]. Multi-valued Decision Diagrams (MDDs) [21], [22] belong to the most efficient representations. These diagrams have been developed as a generalization of Binary Decision Diagrams (BDDs) [23], [24], which are used for representation of Boolean functions. Application of decision diagrams, BDDs as well as MDDs, in reliability analysis have been considered in several papers. Most of the papers focus on their use in quantitative analysis. For example, computation of availability of a BSS and evaluation of importance of the components based on a BDD has been considered in [25], [26]. Algorithms for quantification of reliability of MSSs based on MDDs have been considered in [18], [27]. Works as [18], [28] then have dealt with their use in study of importance of system components for operation of the system.

Decision diagrams can also be used in qualitative reliability analysis, which focuses on detection of scenarios under which specific changes in system performance occur. For example, works [25], [29] have dealt with finding circumstances under which a failure/degradation of a component of a BSS/MSS results in a failure/degradation of the system using a BDD/MDD. In [30], a method for detection of Minimal Cut Sets (MCSs), which represent minimal sets of components whose simultaneous failure results in a failure of a BSS, have been proposed. In this case, it is important to note that any MCS of a BSS can also be expressed in a form of a state vector. This representation is known as a Minimal Cut Vector (MCV), and it defines circumstances under which a repair of any non-working component results in a repair of the whole BSS. Algorithms for their detection based on BDDs can be found in [31].

MCSs and MCVs belong to one of the most important concepts in reliability analysis of BSSs because they are useful not only in qualitative analysis but also in quantitative analysis [4], [32] – [34]. These concepts can also be generalized for MSSs. For example, one of their first generalizations for homogenous MSSs can be found in [35]. Unlike BSSs, where the concept of MCSs is more often used than the concept of MCVs, MCVs are more common in reliability analysis of MSSs than MCSs.

In case of MSSs, a MCV corresponds to a situation in which a minor improvement (i.e., improvement by one state) of any non-perfectly working component results in improvement of the system. Their knowledge can be used, for example, to quantify reliability of a MSS [36], [37] or to evaluate influence of system components or their states on reliability of MSSs [37]. Several algorithms have been proposed for their detection. Most of the algorithms, e.g., [38] – [40], are specially designed for network systems. Another algorithm considered in [41], have been proposed for their identification using a mathematical methodology known as logic differential calculus. In this work we present one more algorithm, which is based on traversing a MDD expressing the structure function of a MSSs. In some sense, the algorithm can be viewed as a generalization of the algorithm proposed in [31] for identification of MCSs of a BSS using BDDs or as a modification of the algorithm proposed in [42] for enumeration of all MCVs of a BSS based on the structure function expressed in a tabular form. Since the algorithm is very simple, it can be used for evaluation of other more sophisticated algorithms, e.g., that considered in [41], which is based on logic differential calculus.

II. Reliability Analysis Based on Minimal Cut Vectors

A. Binary-State Systems

MCVs have an important role in reliability analysis. For their introduction, let us assume that relation $\boldsymbol{y} > \boldsymbol{x}$ between state vectors $\boldsymbol{y} = (y_1, y_2, \dots, y_n)$ and $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ means that $y_i \geq x_i$ for all $i \in \{1,2,\dots,n\}$ and there is at least one i such that $y_i > x_i$. Using this relation, state vector $\boldsymbol{x}$ is a MCV of a BSS if $\phi(\boldsymbol{x}) = 0$ and $\phi(\boldsymbol{y}) = 1$ for any $\boldsymbol{y} > \boldsymbol{x}$ [4]. Specially, if only the first part of this definition holds, i.e., $\phi(\boldsymbol{x}) = 0$, then vector $\boldsymbol{x}$ is known as a cut vector.

The definition of a MCV implies that it corresponds to a situation in which the system is not working but a repair of any non-working component results in making the system functioning. Thank to this special property, MCVs can be used to compute system unavailability, i.e., the probability that the system fails, using the following formula [31]:

$$U = \Pr\{\phi(\boldsymbol{x}) = 0\} = \Pr\left\{\bigcup_{v=1}^{N_C} \{\boldsymbol{x} \leq \mathrm{MCV}_v\}\right\}, \tag{4}$$

where N_C agrees with the number of all MCVs of a BSS, MCV_v denotes the v-th MCV of the system, for $v = 1,2,\dots,N_C$, and event $\{\boldsymbol{x} \le \text{MCV}_v\}$ means that an arbitrary state vector $\boldsymbol{x}$ is less than or equal to the v-th MCV, where relation $\boldsymbol{x} \le \boldsymbol{y}$ between state vectors $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ and $\boldsymbol{y} = (y_1, y_2, \dots, y_n)$ means that $x_i \le y_i$ for all $i \in \{1,2,\dots,n\}$.

System unavailability (4) can next be used in computation of system availability, i.e., the probability that the system is working, as follows [2]:

$$A = \Pr\{\phi(\boldsymbol{x}) = 1\} = 1 - U, \tag{5}$$

or in quantification of importance of component i of the system for operation of the system via Fussell-Vesely's Importance (FVI) [32], [33] in the next manner [34]:

$$\text{FVI}_i = \frac{\Pr\{\exists \text{MCV}(0_i) \in \text{MCVs}; \boldsymbol{x} \le \text{MCV}(0_i)\}}{U}, \tag{6}$$

where MCVs is a set of all MCVs of the system, $\text{MCV}(0_i)$ is a MCV in which $x_i = 0$, and event $\{\exists \text{MCV}(0_i) \in \text{MCVs}; \boldsymbol{x} \le \text{MCV}(0_i)\}$ means that there is at least one MCV with $x_i = 0$ that is greater than or equal to an arbitrary state vector $\boldsymbol{x}$.

A. Multi-State Systems

MCVs are useful not only in reliability analysis of BSSs but also in the analysis of MSSs. Unlike BSSs, MCVs of MSSs are defined with respect to a specific state of the system. In this case, a MCV with respect to state j of a system, for $j = 1,2,\dots,m-1$, is defined as state vector $\boldsymbol{x}$ such that $\phi(\boldsymbol{x}) < j$ and $\phi(\boldsymbol{y}) \ge j$ for any $\boldsymbol{y} > \boldsymbol{x}$ [3]. As in the case of BSSs, any state vector $\boldsymbol{x}$ such that $\phi(\boldsymbol{x}) < j$ is known as a cut vector of a MSS with respect to system state j. According to this definition, a MCV characterizes a situation in which the system is not able to satisfy demand requiring at least performance corresponding to state j of the system but improvement of any non-perfectly working component (a component that is in a state less than $m_i - 1$ for $i \in \{1,2,\dots,n\}$) causes that the system will be able to satisfy this demand.

As in the case of BSSs, MCVs can be used to compute unavailability of the system with respect to state j as follows [37]:

$$U^{\ge j} = \Pr\{\phi(\boldsymbol{x}) < j\} = \Pr\left\{\bigcup_{v=1}^{N_{C^{\ge j}}} \left\{\boldsymbol{x} \le \text{MCV}_v^{\ge j}\right\}\right\}, \quad \text{for } j \in \{1,2,\dots,m-1\}, \tag{7}$$

where, similarly as in the case of BSSs, $N_{C^{\ge j}}$ agrees with the number of all MCVs for state j of the system, $\text{MCV}_v^{\ge j}$ denotes the v-th MCV for state j of the system, for $v = 1,2,\dots,N_{C^{\ge j}}$, and event $\left\{\boldsymbol{x} \le \text{MCV}_v^{\ge j}\right\}$ means that an arbitrary state vector $\boldsymbol{x}$ is less than or equal to the v-th MCV for state j of the system. This formula can be used to compute availability of a MSS, i.e., the probability that the system is able to satisfy demand requiring at least state j of the system in the following way [37]:

$$A^{\ge j} = \Pr\{\phi(\boldsymbol{x}) \ge j\} = 1 - U^{\ge j}, \quad \text{for } j \in \{1,2,\dots,m-1\}. \tag{8}$$

MCVs of MSSs can also be used to analyze importance of a specific component state on a specific state of the system. For this purpose, generalizations of FVI (6) can be used. One of them is the FVI proposed in [43], which allows us to quantify how all states of the i-th system component that are worse than state s contribute to unavailability of the system computed with respect to state j. This FVI is defined as follows:

$$\mathrm{FVI}_{i,s\searrow}^{\geq j} = \frac{\Pr\{\exists \mathrm{MCV}^{\geq j}((s-1)_i) \in \mathrm{MCVs}^{\geq j}; \boldsymbol{x} \leq \mathrm{MCV}^{\geq j}((s-1)_i)\}}{U^{\geq j}}, \tag{9}$$
$$\text{for } s \in \{1,2,\dots,m_i-1\},\ \ j \in \{1,2,\dots,m-1\},$$

where $\mathrm{MCV}^{\geq j}$ is a set of all MCVs for system state j, $\mathrm{MCV}^{\geq j}((s-1)_i)$ is a MCV for system state j in which $x_i = s-1$ and event $\{\exists \mathrm{MCV}^{\geq j}((s-1)_i) \in \mathrm{MCVs}^{\geq j}; \boldsymbol{x} \leq \mathrm{MCV}^{\geq j}((s-1)_i)\}$ means that there is at least one MCV for system state j with $x_i = s-1$ that is greater than or equal to an arbitrary state vector $\boldsymbol{x}$. Slight modification of this measure can be used to study how a minor degradation (degradation by one state) of a given state of a given component contributes to unavailability computed with respect to system state j. This modification has been introduced in [37] and results in the following formula:

$$\mathrm{FVI}_{i,s}^{\geq j} = \frac{\Pr\{\exists \mathrm{MCV}^{\geq j}((s-1)_i) \in \mathrm{MCVs}^{\geq j}; ((s-1)_i, \boldsymbol{x}) \leq \mathrm{MCV}^{\geq j}((s-1)_i)\} p_{i,s-1}}{U^{\geq j}}, \tag{10}$$
$$\text{for } s \in \{1,2,\dots,m_i-1\},\ \ j \in \{1,2,\dots,m-1\},$$

where event $\{\exists \mathrm{MCV}^{\geq j}((s-1)_i) \in \mathrm{MCVs}^{\geq j}; ((s-1)_i, \boldsymbol{x}) \leq \mathrm{MCV}^{\geq j}((s-1)_i)\}$ agree with a statement that there exist at least one MCV for system state j in which $x_i = s-1$ that is greater than or equal to an arbitrary state vector $((s-1)_i, \boldsymbol{x}) = (x_1, x_2, \dots, x_{i-1}, (s-1)_i, x_{i+1}, \dots, x_n)$. This FVI can also be used to compute how a specific component (not only its specific state) contributes to unavailability of the system computed with respect to state j of the system [37].

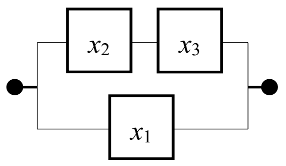
According to the previous paragraphs, MCVs have various usage in reliability analysis. Because of that, their efficient identification belongs to an important issue in reliability analysis. One simple algorithm can be formulated based on the assumption that the structure function is expressed using a decision diagram. Since BSSs can be viewed as a homogeneous MSS with $m = 2$, an algorithm that allows finding all MCVs for state j of a MSS can also be used to find all MCVs of a BSS simply be specifying that $m_1 = m_2 = \cdots = m_n = m = 2$ and $j = 1$. Thus, we will mainly focus on MSSs in what follows.

III. Multi-Valued Decision Diagrams

The structure function of a MSS can be represented in various ways. The most common approaches are truth table, symbolic expression, and MDD (Fig. 1). The truth table explicitly defines system state for each state vector. It is quite simple to analyze system reliability using the truth table, but this representation can only be used for small systems because it has large memory consumption. Another problem of this form is that if we want to estimate system reliability, then we have to investigate all state vectors, which can be quite time-consuming. Symbolic expressions are usually less memory consuming but their processing on a computer is usually quite complicated. A MDD is a compromise between these two representations.

A MDD is a generalization of a BDD, which has been introduced by Lee and Akers in [23], [24] for efficient representation of and manipulation with Boolean functions. A MDD is a directed acyclic graph used for representation of multiple-valued logic [22] and integer functions [21]. It contains one source node (denoted as the root) and several sink nodes. The sink nodes represent values of an integer function represented by the MDD, the root and other internal nodes agree with the variables of the function, and the outgoing edges of the root and each internal node coincide with the possible values of the variables agreeing with the node.

MSS of 3 components



(a) Truth Table

Structure function

$x_2\,x_3$ \ x_1	0	1	2
0 0	0	1	1
0 1	0	1	1
0 2	0	1	1
1 0	0	1	1
1 1	1	2	2
1 2	1	2	3

(b) Symbolic expression

$$\phi(\boldsymbol{x}) = \min\{x_2, x_3\} + \min\{x_1, 1\} + \mathrm{I}(x_1 = 2)\mathrm{I}(x_2 = 1)\mathrm{I}(x_3 = 2),$$

where I(.) is a function that takes value 1 if the argument interpreted as a condition is true, otherwise it takes value 0.

(c) MDD

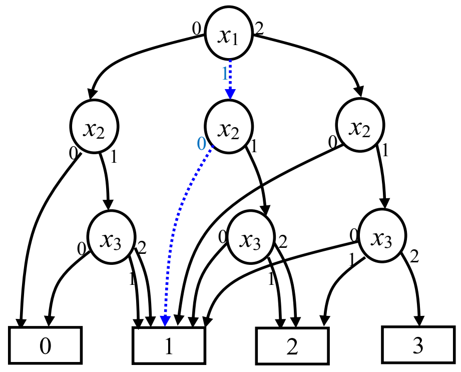


Fig. 1. Various representations of structure function of a MSS composed of 3 components with 4 system states and with components states defined as $m_1 = 3$, $m_2 = 2$, and $m_3 = 3$.

In reliability analysis, a MDD can be used to represent structure function of a MSS [16], [27]. In this case, the sink nodes represent states of the system, therefore, such a MDD contains m sink nodes labeled by numbers from 0 to $m - 1$. The non-sink nodes (the root and all internal nodes) of the MDD correspond to the system components. Each non-sink node that represents the i-th system component, has then m_i outgoing edges labeled by numbers from 0 to $m_i - 1$. These edges correspond to states of the i-th system component. Every path from the root to a sink node, which agrees with state j of the system, represents one or more state vectors. If a path contains each component of the system, then it corresponds to one state vector. If some components are not included in a path, then this path corresponds to more state vectors, i.e., all state vectors, whose elements are present in the path are set according to labels of edges in this path, and elements that are not included in the path can take arbitrary feasible states. For example, a path that does not contain components i_1 and i_2 of the system, whose structure function is represented by the MDD, covers $m_{i_1}m_{i_2}$ different state vectors. If we look at Fig. 1, then we can see that the blue dotted path does not contain variable x_3, which defines state of component 3, therefore, this path covers $m_3 = 3$ different state vectors. These state vectors are (1,0,0), (1,0,1), and (1,0,2), and the structure function takes value 1 for all of them because the blue dotted path, which covers them, ends in sink node labeled by number 1.

A MDD has less demands on memory than the truth table and also it is more convenient for computer implementation than the symbolic expression. Therefore, it can be used for representation of structure function of large systems. Some aspects on evaluation of reliability of a system represented by the MDD can be found in [16], [29], [44]. In this paper, we extend results from those works by introducing an algorithm for enumeration of all MCVs of a MSS.

IV. Algorithm for Enumeration of All Minimal Cut Vectors of a Multi-State System Represented by Multi-Valued Decision Diagram

A. Prepositions

The definition of a MCV introduced in section II implies that a state vector $\boldsymbol{x}$ is a MCV for state j of a MSS if and only if an improvement of any non-perfectly working component, which is in a state defined by the MCV, by one state results in improvement of the system above state $j-1$. In other words, if we use the following notation:

$$\boldsymbol{x}^{(i)} = (x_1, x_2, \dots, x_{i-1}, \min\{x_i + 1, m_i - 1\}, x_{i+1}, \dots, x_n), \tag{11}$$

then it is clear that:

$$\forall \boldsymbol{y}; \boldsymbol{y} > \boldsymbol{x}: \left(\exists i \in \{1,2,\dots,n\}: \boldsymbol{x}^{(i)} \le \boldsymbol{y}\right). \tag{12}$$

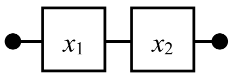
If we denote a set of all MCVs for state j of the system as $\text{MCVs}^{\geq j}$, then we can write:

$$\left(\boldsymbol{x} \in \text{MCVs}^{\geq j}\right) \Leftrightarrow \left(\forall i \in \{1,2,\dots,n\}; \boldsymbol{x}^{(i)} \neq \boldsymbol{x}: \phi\left(\boldsymbol{x}^{(i)}\right) \geq j\right). \tag{13}$$

Formula (13) implies if want to check whether a state vector is a MCV, then we have to compare it with at most n other state vectors. This formula can be used for formulation of algorithms for finding all MCVs. In [42], it has been used for introducing an algorithm that assumes that the structure function is represented by the truth table. In this paper, we use it for development of a simple algorithm for enumeration of all MCVs of a MSS based on the assumption that the structure function is defined by a MDD.

One important fact that also results from the definition of a MCV of a MSS is that one state vector can be a MCV for several states of the system. This situation occurs when degradation of one or more system components by one state can result in a decrease in system performance by more than one state. An example of such a system is shown in Fig. 2. As one can see, state vector $(0,2)$ in this system is a MCV for states 1 and 2 of the system.

MSS of 2 components in which $m = 3$, $m_1 = 3$, and $m_2 = 2$



Structure function

x_2 \ x_1	0	1
0	0	0
1	0	1
2	0	2

Fig. 2. Example of a simple series system in which a minor degradation of one component can result in degradation of the system by more than one state.

The definition of a MCV and the assumption of system coherency imply if state vector $\boldsymbol{x}$ is a MCV for states j_1 and j_2 of the system, where $0 < j_1 < j_2 \leq m-1$, then it is also a MCV for all states of the system between values j_1 and j_2. So, we only need to know the minimal and maximal states of the system for which state vector $\boldsymbol{x}$ is a MCV. These two levels can be found by using the following statements:

S1) if state vector $\boldsymbol{x}$ is a MCV for a state of the system, then the minimal state of the system for which it is a MCV agrees with $\phi(\boldsymbol{x}) + 1$,

S2) if state vector $\boldsymbol{x}$ is a MCV for a state of the system, then the maximal state of the system for which it is a MCV is defined as $\min\left\{\phi\left(\boldsymbol{x}^{(i)}\right) \middle| i = 1,2,\dots,n \text{ AND } \boldsymbol{x}^{(i)} \neq \boldsymbol{x}\right\}$.

Based on these two statements and the ideas presented above we can formulate the next algorithm that allows us to enumerate all MCVs for all relevant states of a MSS based on the assumption that the structure function of the MSS is represented by a MDD.

B. Algorithm

The algorithm for finding all MCVs of a MSS has two phases. The basic idea of the first phase lies in finding all paths from the root to sink nodes in the MDD for all system states except state $m-1$ (the definition of a MCV implies that a state vector for which the system is in state $m-1$ cannot be a MCV). As we mentioned earlier, each path in a MDD corresponds to one or more state vectors. If there are more state vectors that are covered by the considered MDD path, then the definition of a MCV introduced in section II implies that only the maximal state vector can be a MCV. All these maximal state vectors identified during traversing the MDD are candidates for MCVs of a MSS.

In the second phase, we only check which candidates are real MCVs and find minimal and maximal system states, which are associated with given MCVs. To do this, we use the property (13) of a MCV and statements S1) and S2). This second phase can also be performed during the first phase, i.e., during the MDD traversal. In this case, we traverse the MDD from sink nodes to the root and check whether the maximal state vector corresponding to a path identified during the traversing meets property (13). The algorithm based on this idea can be formalized as follows:

1. Set the Current Node (CN) to sink node with label 0. Set variable $j=0$. (Value $j+1$ represents the minimal value of system state for which we enumerate MCVs).
2. If $j=m-1$, then STOP because all MCVs has been found.
3. Initialize state vector $\boldsymbol{x}=(m_1-1, m_2-1, \dots, m_n-1)$ and variable $j_{max}=m-1$. (State vector $\boldsymbol{x}$ represents a candidate for a MCV and variable j_{max} corresponds to the maximal state of the system for which $\boldsymbol{x}$ can be a MCV.)
4. If the CN is the root of the MDD, then state vector $\boldsymbol{x}$ is a MCV for all system states from $j+1$ to j_{max}.
5. From the CN move to a Parent Node (PN) that has not yet been visited from the CN for the current state vector $\boldsymbol{x}$ using the edge with maximal value of the label and set the value on a position in the state vector $\boldsymbol{x}$, which corresponds to the PN, to the label of this edge.
    a. If there is no unvisited PN, then set the value on a position in the state vector $\boldsymbol{x}$, which corresponds to the CN, to the maximum (i.e., if the CN coincides with the i-th component, then use m_i-1), set variable j_{max} to the value that has been valid in the previous CN, set the CN to the previous CN and GO TO STEP 5. If the previous CN does not exist (the CN is the sink node with label j), then increment j by 1 and GO TO STEP 2.
    b. If the PN has no edge that is labeled by a value greater than the corresponding value in the state vector $\boldsymbol{x}$, then GO TO STEP 8.
    c. If the PN has edges that are labeled by values greater than the corresponding value in the state vector $\boldsymbol{x}$ (the corresponding component can be in better states), then increment the corresponding value in this state vector by 1 and using the state vector $\boldsymbol{x}$ check if the path from the PN ends in a sink node with label that is greater than j (we denote this path as *P_PN_SiN*). Then, decrement the corresponding value in the state vector $\boldsymbol{x}$ by 1 (after this we get the original state vector $\boldsymbol{x}$).
6. If the path *P_PN_SiN* ends in a sink node with label j, then no state vector covered by the path from the root through the PN and the CN to the sink node j (we denote this

path as *P_R_PN_CN_SiN*) can be a MCV. In this case set the value in state vector $\boldsymbol{x}$, which corresponds to the CN, to the maximum and GO TO STEP 5.

7. If the path *P_PN_SiN* ends in a sink node with label greater than j, then some state vectors covered by the path *P_R_PN_CN_SiN* can be MCVs. In this case set variable j_{max} to the minimum from j_{max} and the label of the sink node lying at the end of the path *P_PN_SiN*.
8. Set the CN to the PN. GO TO STEP 4.

C. The Graphical Interpretation of the Algorithm

In this section, we illustrate how the algorithm proposed in the previous section works on the MDD representing structure function of the system depicted in Fig. 1.

1. We begin in sink node 0: set the CN to sink node 0 and j to 0.
2. Variable j is less than 3 ($m = 4$), so we go to the next step.
3. Set state vector $\boldsymbol{x} = (2,1,2)$ and variable $j_{max} = 3$.
4. The CN is not the root, so we continue to the next step.
5. The CN has two unvisited parent nodes for the current state vector $\boldsymbol{x} = (2,1,2)$ that correspond to variables x_2 and x_3, so we set the PN to the node corresponding to variable x_2. Since the edge with the maximal value of label from the PN to the CN is labeled by number 0, we set $\boldsymbol{x} = (2,0,2)$. (Look at the blue dotted line denoted by number 5 in Fig. 3.)
    a. An unvisited PN has existed (it corresponds to variable x_2), so we continue to the next step.
    b. The PN corresponding to variable x_2 has an edge that is labeled by a value greater than value 0 (the value at the position corresponding to variable x_2 in state vector $\boldsymbol{x} = (2,0,2)$), so we go to the next step.
    c. We set $\boldsymbol{x} = (2,1,2)$ and check if the path *P_PN_SiN* from the PN, which corresponds to variable x_2, ends in a sink node that is greater than 0. This is true. Next, we set $\boldsymbol{x} = (2,0,2)$.
6. The path *P_PN_SiN* ends in a sink node with label 1, which is greater than 0 (look at the red solid line denoted by number 6 in Fig. 3), so we continue to the next step.
7. We set $j_{max} = \min\{3,1\} = 1$ and continue to the next step.
8. We set CN = PN (the CN now corresponds to variable x_2) and continue to step 4.

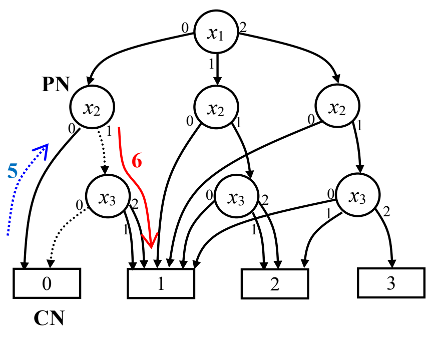


Fig. 3. The first part of the algorithm running on the MDD depicted in Fig. 1.

4. The CN corresponding to variable x_2 is not the root, so we continue to the next step.

5. The CN has one unvisited PN (it corresponds to variable x_1) for the current state vector $\boldsymbol{x} = (2,0,2)$, so we set the PN to it and $\boldsymbol{x} = (0,0,2)$ because the edge with the maximal value of label from the PN to the CN is labeled by number 0. (Look at the blue dotted line denoted by value 5 in Fig. 4.)
    a. An unvisited PN has existed (it corresponds to variable x_1), so we continue to the next step.
    b. The PN corresponding to variable x_1 has an edge that is labeled by value greater than value 0, so we go to the next step.
    c. We set $\boldsymbol{x} = (1,0,2)$ and check if the path *P_PN_SiN* from the PN, which corresponds to variable x_1, according to values stored in state vector $\boldsymbol{x}$ ends in a sink node greater than 0. This is true. Next, we set $\boldsymbol{x} = (0,0,2)$.
6. The path *P_PN_SiN* does not end in a sink node with label 0 (look at the red solid line denoted by number 6 in Fig. 4), so we go to the next step.
7. The path *P_PN_SiN* ends in a sink node with label 1, so it means that some state vectors covered by the path *P_R_PN_CN_SiN*, can be MCVs. Because of that, we set $j_{max} = \min\{1,1\} = 1$ and continue to the next step.
8. We set CN = PN (the CN now corresponds to variable x_1) and continue to step 4.

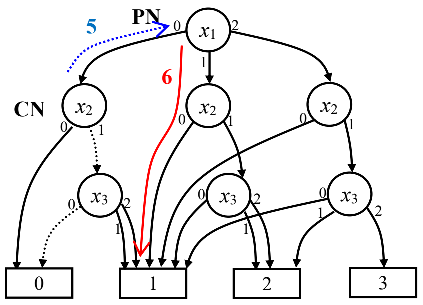


Fig. 4. The second part of the algorithm running on the MDD depicted in Fig. 1.

4. The CN corresponding to variable x_1 is the root, so the current state vector $\boldsymbol{x} = (0,0,2)$ is a MCV for state 1 of the system (i.e., for system states from $j + 1 = 1$ to $j_{max} = 1$).
5. There is no unvisited PN for the current state vector $\boldsymbol{x} = (0,0,2)$, so we continue to the next sub-step.
    a. The CN corresponds to variable x_1 whose maximal value is 2, therefore, we set $\boldsymbol{x} = (2,0,2)$. Next we set $j_{max} = 1$ (this value has been valid in the previous CN); CN = "the previous CN, which corresponds to variable x_2". Next we go to the beginning of step 5. (Look at the red solid line denoted as 5.a in Fig. 5.)

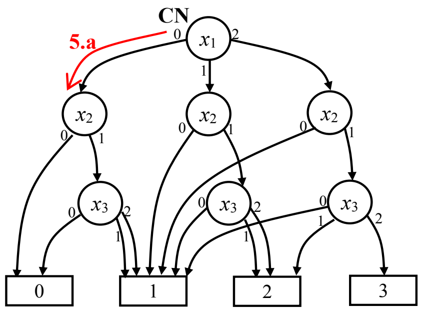


Fig. 5. The third part of the algorithm running on the MDD depicted in Fig. 1.

5. The CN corresponding to variable x_2 has no unvisited PN for the current state vector $\boldsymbol{x} = (2,0,2)$, so we continue to the next sub-step.
    a. The CN corresponds to variable x_2 whose maximal value is 1, therefore, we set $\boldsymbol{x} = (2,1,2)$. During the bottom-up traversing, the previous CN agreed with sink node 0, so we set $j_{max} = 3$ and CN = "sink node 0". Next we go to the beginning of step 5. (Look at the red solid line denoted as 5.a in Fig. 6.)

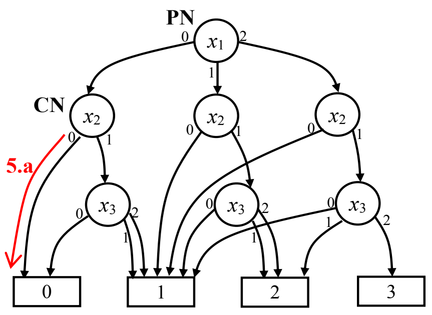


Fig. 6. The fourth part of the algorithm running on the MDD depicted in Fig. 1.

5. The CN (sink node 0) has only one unvisited node (it corresponds to variable x_3) for the current state vector $\boldsymbol{x} = (2,1,2)$, so we set the PN to that node and $\boldsymbol{x} = (2,1,0)$ because the edge with the maximal value of label from the PN to the CN is labeled by number 0. (Look at the blue dotted line denoted by number 5 in Fig. 7.)
    a. An unvisited PN has existed (it corresponds to variable x_2), so we continue to the next step.
    b. The PN corresponding to variable x_3 has an edge that is labeled by a value greater than value 0 (the value at the position corresponding to variable x_3 in state vector $\boldsymbol{x} = (2,1,0)$), so we go to the next step.
    c. We set $\boldsymbol{x} = (2,1,1)$ and check if the path *P_PN_SiN* from the PN, which corresponds to variable x_3, ends in a sink node that is greater than 0. This is true. Next, we set $\boldsymbol{x} = (2,1,0)$.
6. The path *P_PN_SiN* ends in a sink node with label 1, which is greater than 0 (look at the red solid line denoted by number 6 in Fig. 7), so we continue to the next step.
7. We set $j_{max} = \min\{3,1\} = 1$ and continue to the next step.
8. We set CN = PN (the CN now corresponds to variable x_3) and continue to step 4.

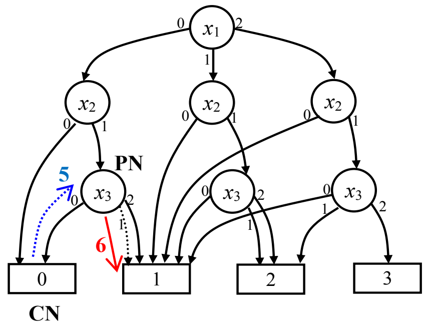


Fig. 7. The fifth part of the algorithm running on the MDD depicted in Fig. 1.

4. The CN corresponding to variable x_3 is not the root, so we continue to the next step.
5. The CN has one unvisited PN (it corresponds to variable x_2) for the current state vector

$\boldsymbol{x} = (2{,}1{,}0)$, so we set the PN to it and $\boldsymbol{x} = (2{,}1{,}0)$ because the edge with the maximal value of label from the PN to the CN is labeled by number 1. (Look at the blue dotted line denoted by value 5 in Fig. 8.)

   a. An unvisited PN has existed (it corresponds to variable x_2), so we continue to the next step.

   b. The PN corresponding to variable x_2 has no edge that is labeled by value greater than value 1, so we continue to step 8.

8. We set CN = PN (the CN now corresponds to variable x_2) and continue to step 4.

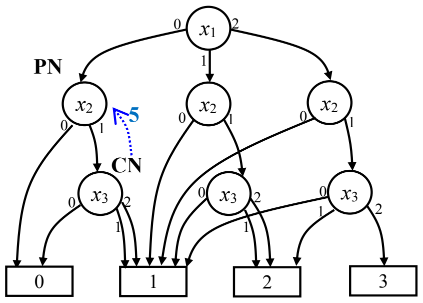


Fig. 8. The sixth part of the algorithm running on the MDD depicted in Fig. 1.

4. The CN corresponding to variable x_2 is not the root, so we continue to the next step.
5. The CN has one unvisited PN (it corresponds to variable x_1) for the current state vector $\boldsymbol{x} = (2{,}1{,}0)$, so we set the PN to it and $\boldsymbol{x} = (0{,}1{,}0)$ because the edge with the maximal value of label from the PN to the CN is labeled by number 0. (Look at the blue dotted line denoted by value 5 in Fig. 9.)

   a. An unvisited PN has existed (it corresponds to variable x_1), so we continue to the next step.

   b. The PN corresponding to variable x_1 has an edge that is labeled by value greater than value 0, so we go to the next step.

   c. We set $\boldsymbol{x} = (1{,}1{,}0)$ and check if the path *P_PN_SiN* from the PN, which corresponds to variable x_1, according to values stored in state vector $\boldsymbol{x}$ ends in a sink node greater than 0. This is true. Next, we set $\boldsymbol{x} = (0{,}1{,}0)$.

6. The path *P_PN_SiN* does not end in a sink node with label 0 (look at the red solid line denoted by number 6 in Fig. 9), so we go to the next step.

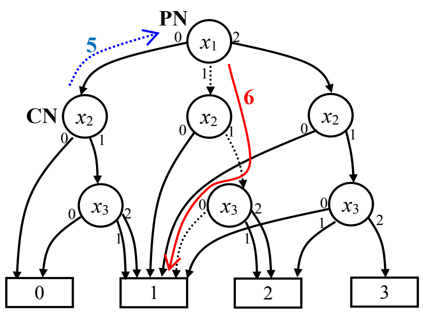


Fig. 9. The seventh part of the algorithm running on the MDD depicted in Fig. 1.

7. The path *P_PN_SiN* ends in a sink node with label 1, so it means that some state vectors covered by the path *P_R_PN_CN_SiN*, can be MCVs. Because of that, we set $j_{max} = \min\{1{,}1\} = 1$ and continue to the next step.

8. We set CN = PN (the CN now corresponds to variable x_1) and continue to step 4.
4. The CN corresponding to variable x_1 is the root, so state vector $\boldsymbol{x} = (0,1,0)$ is a MCV for state 1 of the system (i.e., for system states from $j + 1 = 1$ to $j_{max} = 1$).
5. There is no unvisited PN for the current state vector $\boldsymbol{x} = (0,1,0)$, so we continue to the next sub-step.
   a. The CN corresponds to variable x_1 whose maximal value is 2, therefore, we set $\boldsymbol{x} = (2,1,0)$. Next we set $j_{max} = 1$ (this value has been valid in the previous CN); CN = "the previous CN, which corresponds to variable x_2". Next we go to the beginning of step 5. (Look at the red solid line denoted as 5.a in Fig. 10.)

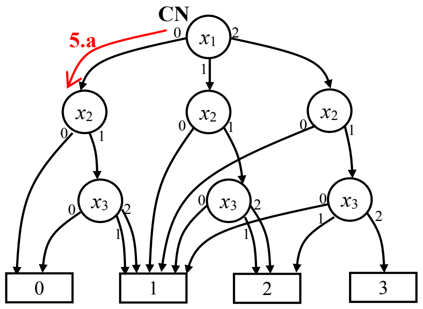


Fig. 10. The eight part of the algorithm running on the MDD depicted in Fig. 1.

6. The CN corresponding to variable x_2 has no unvisited PN for the current state vector $\boldsymbol{x} = (2,1,0)$, so we continue to the next sub-step.
   a. The CN corresponds to variable x_2 whose maximal value is 1, therefore, we set $\boldsymbol{x} = (2,1,0)$. During the bottom-up traversing, the previous CN agreed with the node corresponding to variable x_3, so we set $j_{max} = 1$ and CN = "the previous CN, which corresponds to variable x_3". Next we go to the beginning of step 5. (Look at the red solid line denoted as 5.a in Fig. 11.)

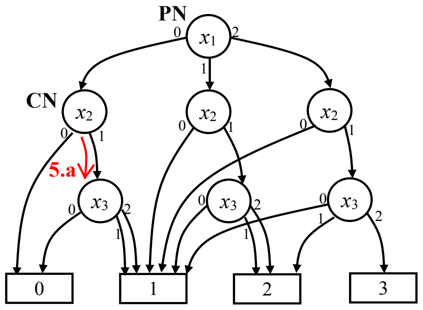


Fig. 11. The ninth part of the algorithm running on the MDD depicted in Fig. 1.

5. The CN corresponding to variable x_3 has no unvisited PN for the current state vector $\boldsymbol{x} = (2,1,0)$, so we continue to the next sub-step.
   a. The CN corresponds to variable x_3 whose maximal value is 2, therefore, we set $\boldsymbol{x} = (2,1,2)$. During the bottom-up traversing, the previous CN agreed with the sink node 0, so we set CN = "sink node 0". In those time, j_{max} was 3, therefore, we set $j_{max} = 3$ and continue to the beginning of step 5. (Look at the red solid line denoted as 5.a in Fig. 12.)
5. The CN corresponding to sink node 0 has no unvisited PN for the current state vector $\boldsymbol{x} = (2,1,2)$, so we continue to the next sub-step.

a. There does not exist any previous CN, so we increment j by 1 (i.e., $j = 1$) and continue to step 2. In this phase, we have all MCVs for state 1 of the system. In the following steps, we gain MCVs for other states of the system (i.e., for 2 and 3).

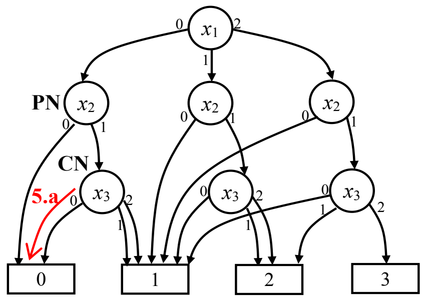


Fig. 12. The tenth part of the algorithm running on the MDD depicted in Fig. 1.

Using the algorithm described above, we can find that the system has two MCVs for state 1 (state vectors $(0,0,2)$ and $(0,1,0)$), three for state 2 (state vectors $(0,1,2)$, $(2,0,2)$, and $(2,1,0)$) and three for state 3 (state vectors $(1,1,2)$, $(2,0,2)$, and $(2,1,1)$). These MCVs are same as those enumerated based on the truth table shown in Fig. 1 using the generalization of the algorithm proposed in [42] for MSSs.

V. Conclusion

MCVs play an important role in reliability analysis. They are useful in computation of system unavailability or availability, in investigation of importance of the system components for operation of the system, or in planning system maintenance. Because of that, one of the key tasks of reliability analysis is their efficient identification. In this paper, we presented a simple algorithm for their enumeration based on the assumption that the structure function of the system is represented by a MDD. This algorithm traverses the MDD and identifies candidates for MCVs. During the traversing, each candidate is checked based on property (13). If it meets this property, then the candidate is a MCV. However, as we saw in the graphical illustration of the algorithm, some nodes of the MDD can be visited several times. In the future work, we would like to modify this algorithm in such a way that all nodes of the MDD will be visited at most once. This can result in more efficient algorithm for enumeration of MCVs.

Acknowledgment

This publication was realized with the support of the „Operational Program Research and Innovation in the frame of the project: ICT for smart society, code ITMS2014 +: 313011T462, co-financed by the European Regional Development Fund".

References

[1] E. Zio, “Reliability engineering: Old problems and new challenges,” *Reliability Engineering & System Safety*, vol. 94, no. 2, pp. 125–141, Feb. 2009.
[2] M. Rausand and A. Høyland, *System Reliability Theory, 2nd ed*. Hoboken, NJ: John Wiley & Sons, Inc., 2004.
[3] A. Lisnianski and G. Levitin, *Multi-state System Reliability. Assessment, Optimization and Applications*. Singapore, SG: World Scientific, 2003.
[4] W. Kuo and X. Zhu, *Importance Measures in Reliability, Risk, and Optimization: Principles and Applications*. Chichester, UK: Wiley, 2012.
[5] G. Levitin, *The Universal Generating Function in Reliability Analysis and Optimization*. London: Springer-Verlag, 2005.
[6] D. J. Smith, *Reliability, Maintainability and Risk: Practical Methods for Engineers, Ninth Edition*. Oxford, UK: Butterworth-Heinemann, 2017.

[7] E. N. Zaitseva and V. G. Levashenko, “Importance analysis by logical differential calculus,” *Automation and Remote Control*, vol. 74, no. 2, pp. 171–182, Feb. 2013.

[8] Y. Watanabe, T. Oikawa, and K. Muramatsu, “Development of the DQFM method to consider the effect of correlation of component failures in seismic PSA of nuclear power plant,” *Reliability Engineering & System Safety*, vol. 79, no. 3, pp. 265–279, Mar. 2003.

[9] B. Nystrom, L. Austrin, N. Ankarback, and E. Nilsson, “Fault tree analysis of an aircraft electric power supply system to electrical actuators,” in *International Conference on Probabilistic Methods Applied to Power Systems, 2006. PMAPS 2006*, 2006, pp. 1–7.

[10] M. Kvassay, E. Zaitseva, V. Levashenko, and J. Kostolny, “Reliability analysis of multiple-outputs logic circuits based on structure function approach,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 3, pp. 398–411, 2017.

[11] E. Zio, *An Introduction to the Basics of Reliability and Risk Analysis*. Singapore, SG: World Scientific Publishing Co. Inc., 2007.

[12] P. Praks, V. Kopustinskas, and M. Masera, “Probabilistic modelling of security of supply in gas networks and evaluation of new infrastructure,” *Reliability Engineering & System Safety*, vol. 144, pp. 254–264, Dec. 2015.

[13] E. Zaitseva, V. Levashenko, and M. Rusin, “Reliability analysis of healthcare system,” in *2011 Federated Conference on Computer Science and Information Systems, FedCSIS 2011*, 2011, pp. 169–175.

[14] P. Rusnak, P. Sedlacek, A. Forgac, O. Illiashenko, and V. Kharchenko, “Structure function based methods in evaluation of availability of healthcare system,” in *2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 2019, pp. 13–18.

[15] M. Kvet, E. Krsak, and K. Matiasko, “Temporal database architecture enhancements,” in *2018 22nd Conference of Open Innovations Association (FRUCT)*, 2018, pp. 121–130.

[16] E. Zaitseva and V. Levashenko, “Multi-state system analysis based on multiple-valued decision diagram,” *Journal of Reliability and Statistical Studies*, vol. 5, no. Special, pp. 107–118, 2012.

[17] A. Forgac, J. Rabcan, E. Zaitseva, and I. Lukyanchuk, “Construction of the structure function of multi-state system based on incompletely specified data,” in *Contemporary Complex Systems and Their Dependability*, W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, and J. Kacprzyk, Eds. Cham: Springer International Publishing, 2019, pp. 184–194.

[18] E. Zaitseva and V. Levashenko, “Multiple-valued logic mathematical approaches for multi-state system reliability analysis,” *Journal of Applied Logic*, vol. 11, no. 3, pp. 350–362, Sep. 2013.

[19] M. Kvassay, P. Rusnak, and J. Rabcan, “Time-Dependent Analysis of Series-Parallel Multistate Systems Using Structure Function and Markov Processes,” in *Advances in System Reliability Engineering*, M. Ram and P. Davim, Eds. Academic Press, 2019, pp. 131–165.

[20] A. Kaufmann, D. Grouchko, and R. Cruon, *Mathematical Models for the Study of the Reliability of Systems*. New York, NY: Academic Press, 1977.

[21] T. Sasao and M. Fujita, Eds., *Representations of Discrete Functions*. Boston, MA: Springer US, 1996.

[22] S. Yanushkevich, D. Michael Miller, V. Shmerko, and R. Stankovic, *Decision Diagram Techniques for Micro- and Nanoelectronic Design Handbook*, vol. 2. Boca Raton, FL: CRC Press, 2005.

[23] C. Y. Lee, “Representation of switching circuits by binary-decision programs,” *The Bell System Technical Journal*, vol. 38, no. 4, pp. 985–999, 1959.

[24] S. B. Akers, “Binary decision diagrams,” *IEEE Transactions on Computers*, vol. C-27, no. 6, pp. 509–516, Jun. 1978.

[25] E. Zaitseva, V. Levashenko, and J. Kostolny, “Importance analysis based on logical differential calculus and Binary Decision Diagram,” *Reliability Engineering & System Safety*, vol. 138, pp. 135–144, Jun. 2015.

[26] M. Kvassay, E. Zaitseva, V. Levashenko, and J. Kostolny, “Binary decision diagrams in reliability analysis of standard system structures,” in *2016 International Conference on Information and Digital Technologies (IDT)*, 2016, pp. 164–172.

[27] Liudong Xing and Yuanshun Dai, “A new decision-diagram-based method for efficient analysis on multistate systems,” *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 3, pp. 161–174, Jul. 2009.

[28] A. Shrestha, L. Xing, and D. W. Coit, “Multi-state component importance analysis using multi-state multi-valued decision diagrams,” in *2009 8th International Conference on Reliability, Maintainability and Safety*, 2009, pp. 99–103.

[29] M. Kvassay, E. Zaitseva, J. Kostolny, and V. Levashenko, “Reliability analysis of multi-state systems based on tools of multiple-valued logic,” in *IEEE EUROCON 2015 – International Conference on Computer as a Tool (EUROCON)*, 2015, pp. 1–6.

[30] M.-L. Rebaiaia and D. Ait-Kadi, “A new technique for generating minimal cut sets in nontrivial network,” *AASRI Procedia*, vol. 5, pp. 67–76, Jan. 2013.

[31] M. Kvassay, P. Rusnak, E. Zaitseva, and J. Kostolny, “Application of logic differential calculus and binary decision diagrams in detection of minimal cut vectors,” in *Proceedings of the 29th European Safety and Reliability Conference, ESREL 2019*, 2019, pp. 802–809.

[32] W. E. Vesely, “A time-dependent methodology for fault tree evaluation,” *Nuclear Engineering and Design*, vol. 13, no. 2, pp. 337–360, Aug. 1970.

[33] J. B. Fussell, “How to hand-calculate system reliability and safety characteristics,” *IEEE Transactions on Reliability*, vol. R-24, no. 3, pp. 169–174, Aug. 1975.

[34] M. Kvassay, V. Levashenko, and E. Zaitseva, “Analysis of minimal cut and path sets based on direct partial Boolean derivatives,” *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 230, no. 2, pp. 147–161, Apr. 2016.

[35] D. A. Butler, “A complete importance ranking for components of binary coherent systems, with extensions to multi-state systems,” *Naval Research Logistics Quarterly*, vol. 26, no. 4, pp. 565–578, Dec. 1979.

[36] M. Forghani-elahabad and N. Kagan, “An approximate approach for reliability evaluation of a multistate flow network in terms of minimal cuts,” *Journal of Computational Science*, vol. 33, pp. 61–67, 2019.

[37] M. Kvassay, E. Zaitseva, and V. Levashenko, “Minimal cut and minimal path vectors in reliability analysis of binary- and multi-state systems,” in *CEUR Workshop Proceedings*, 2017, vol. 1844, pp. 713–726.

[38] W.-C. Yeh, “A fast algorithm for searching all multi-state minimal cuts,” *IEEE Transactions on Reliability*, vol. 57, no. 4, pp. 581–588, Dec. 2008.

[39] M. Forghani-elahabad and N. Mahdavi-Amiri, “A new efficient approach to search for all multi-state minimal cuts,” *IEEE Transactions on Reliability*, vol. 63, no. 1, pp. 154–166, Mar. 2014.

[40] Y. Niu and X. Xu, “A new solution algorithm for the multistate minimal cut problem,” *IEEE Transactions on Reliability*, pp. 1–13, 2019.

[41] M. Kvassay and J. Kostolny, “Evaluation of algorithms for identification of minimal cut vectors and minimal path vectors in multi-state systems,” *Komunikacie*, vol. 17, no. 4, pp. 8–14, 2015.

[42] M. Kvassay, “An algorithm for finding all minimal cut vectors in reliability analysis,” in *TRANSCOM 2013: 10-th European Conference of Young Research and Scientific Workers*, 2013, pp. 57–60.

[43] M. Kvassay, E. Zaitseva, and V. Levashenko, “Minimal cut sets and direct partial logic derivatives in reliability analysis,” in *Safety and Reliability: Methodology and Applications - Proceedings of the European Safety and Reliability Conference, ESREL 2014*, 2015, pp. 241–248.

[44] E. Zaitseva, V. Levashenko, J. Kostolny, and M. Kvassay, “A multi-valued decision diagram for estimation of multi-state system,” in *Eurocon 2013*, 2013, pp. 645–650.