Hindawi Publishing Corporation
Journal of Quality and Reliability Engineering
Volume 2013, Article ID 532350, 14 pages
http://dx.doi.org/10.1155/2013/532350

Research Article

A Heuristic Methodology for Efficient Reduction of Large Multistate Event Trees

Eftychia C. Marcoulaki

System Reliability and Industrial Safety Laboratory, National Centre for Scientific Research "Demokritos", P.O. Box 60228, Agia Paraskevi, 15310 Athens, Greece

Correspondence should be addressed to Eftychia C. Marcoulaki; emarcoulaki@ipta.demokritos.gr

Received 31 May 2012; Revised 22 August 2012; Accepted 26 September 2012

Academic Editor: Nikolaos E. Limnios

This work proposes a new methodology for the management of event tree information used in the quantitative risk assessment of complex systems. The size of event trees increases exponentially with the number of system components and the number of states that each component can be found in. Their reduction to a manageable set of events can facilitate risk quantification and safety optimization tasks. The proposed method launches a deductive exploitation of the event space, to generate reduced event trees for large multistate systems. The approach consists in the simultaneous treatment of large subsets of the tree, rather than focusing on the given single components of the system and getting trapped into guesses on their structural arrangement.

1. Introduction

For a given system, the scope of quantitative risk assessment is to investigate the circumstances giving rise to different modes of system operation and to quantify the risk for each operation mode. A system can be comprised of hardware, software, humans, or organizational components [1]. Each component can be found in various states of operation, leading to multiple modes of failure and normal operation for the overall system. Once this mapping of component states to system outcomes is known, it is theoretically possible to quantify the risks for different operation modes to occur, given the occurrence probabilities for all the component states [2, 3].

The computational effort and the memory requirements for risk evaluations increase exponentially as the system components and the number of component states increases. Exact calculations for binary systems are achieved faster by employing binary decision diagrams to effectively organize the evaluation procedure [4]. Since the logic behind multistate systems is not Boolean, multistate behavior cannot be represented by binary models without introducing additional variables and constraints [5]. Rocco and Muselli [6] developed a methodology based on machine-learning and hamming clustering to address multistate systems and any success criterion. The required computational resources can be reduced using approximate risk estimations [7] or criticality analysis [8]. Event trees represent the combination of component states leading to each mode of system operation. Quantification of event trees enables faster exact risk evaluation [2] and is not limited to binary or two-terminal systems, it is, however, very computationally intensive. Clearly, implementation of event tree quantification to systems with many components in multiple states needs to be preceded by substantial reduction of the number of tree branches.

This work develops an algorithm that can efficiently exploit a large event space and generate a reduced event tree. It is assumed that every real system has an intrinsic logic behind the assignment of system outcomes to the system events. A simple and general methodology is suggested to robustly extract this knowledge, by exploiting the system outcome space information as this is stored in a table listing all the possible event combinations and their associated final system outcomes. The proposed algorithm is not biased by any prior information on the functionalities of the system in structural or algebraic form and seeks to acquire knowledge on the system logic and encapsulate it in the reduced tree. The algorithm can be generally applied to any given system and is not affected by the way that the supplied data may be organized or sorted. The procedure is deductive, starting from the entire event table and systematically organizing the system events in a set of clusters. The final set of clusters can be translated back to an event tree that is significantly reduced compared to original outcome space information supplied to the algorithm.

The paper is organized as follows: Section 2 presents some very basic system definitions used in the system representation and operations presented in Section 3. In Section 3 a suitable system representation is defined, using sets of events and based on the concepts of Cartesian products. Section 4 describes a set of clustering and declustering operations to be applied on the event sets. Section 5 discusses the implementation of the proposed developments into an algorithmic procedure. Section 6 presents an illustration example and a large case study to demonstrate the proposed methodology. Section 7 concludes the work.

2. Basic System Definitions

The system considered here is comprised of *blocks*. Each block can be found in various states, and the system response (or output) at every given instance of time, t, depends on the state that the blocks occupy at t. Each block relates to a component, a set of components or a part of a component of the system, regardless of physical conventions and according to the choices in the system modeling. The basic definitions are taken from Papazoglou [9, 10] where the blocks were interconnected to form functional block diagrams. In the present work, the blocks are stripped of their networking functionalities, and the definitions are simplified accordingly.

2.1. System Blocks and Their States. Consider a system of K-independent *blocks*. Each block b_k, $k \in \{1, 2, \dots, K\}$, can be found in various *internal states* during the system operation period.

Let the *state set of block* b_k, denoted as $\mathbf{S}_k = \{s_k^1, s_k^2, \dots, s_k^{K_{S_k}}\}$, be a partition over the possible instances of b_k, where $s_k^{i_k}$ denotes the i_kth state of block b_k, and K_{S_k} denotes the number of elements in $\mathbf{S}_k$, thus K_{S_k} is the cardinality of S_k, $|S_k|$.

It is assumed that, at every given time instance within the period of system operation, b_k is found in exactly one state (e.g., at 94% of maximum production level), and this state relates to exactly one member of $\mathbf{S}_k$ (e.g., $s_k^2 =$ "at least 90% of maximum production level").

2.2. Event Definitions. A *basic event* is an instance of a single system block according to the state set partition of this block. A basic event for block b_k is denoted as $\{s_k^{i_k}\}$, $s_k^{i_k} \in \mathbf{S}_k$.

A *joint event* $\{s_{k_1}^{i_{k_1}}, s_{k_2}^{i_{k_2}}, \dots, s_{k_m}^{i_{k_m}}\}$ is the combination of more than one basic events taking place in m different blocks of the system $b_{k_1}, b_{k_2}, \dots, b_{k_m}$. Note that, the set $\{s_{k_1}^{i_{k_1}}, s_{k_2}^{i_{k_2}}, \dots, s_{k_m}^{i_{k_m}}\}$ is the result of the Cartesian product $\{s_{k_1}^{i_{k_1}}\} \times \{s_{k_2}^{i_{k_2}}\} \times \cdots \times \{s_{k_m}^{i_{k_m}}\}$.

A *complete joint event* $\hat{e}$ is defined here as a joint event over all the system blocks $\{s_1^{i_1}, s_2^{i_2}, \dots, s_K^{i_K}\}$. For simplicity, the term *event* is used here instead of complete joint event.

Note that, the above event definitions are simplified compared to Papazoglou [10] where blocks had functionalities not considered here.

2.3. Event Space, Subspaces, and Event Partitions. Let the *system event space* $\mathbf{E}$ be the set of all the possible complete joint events $\hat{e}$. Then $\mathbf{E} = \{\widehat{e_1}, \widehat{e_2}, \dots, \widehat{e_{K_E}}\}$, where K_E denotes the number of elements $\mathbf{E}$, thus $K_E = |E|$. Note that, $\mathbf{E}$ is the result of the Cartesian product $\mathbf{S}_1 \times \mathbf{S}_2 \times \cdots \times \mathbf{S}_K$, therefore $|\mathbf{E}| = \prod_{k=1}^{K} |\mathbf{S}_k|$.

Consider a nonempty subspace $\mathbf{A}$ of the system event space $\mathbf{A} \subseteq \mathbf{E} : \mathbf{A} \neq \varnothing$. Let $\mathbf{Q}(\mathbf{A})$ denote a *partition* applied over $\mathbf{A}$, composed of $K_{\mathbf{Q}(\mathbf{A})}$ disjoint subspaces of $\mathbf{A}$ denoted by $\mathbf{q}_i^{Q}(\mathbf{A})$, $i \in \{1, 2, \dots, K_{\mathbf{Q}(\mathbf{A})}\}$.

2.4. Event Table. Let $\mathbf{R} = \{r_1, r_2, \dots, r_{K_\mathbf{R}}\}$ denote the set of the all possible *system outcomes* and $K_\mathbf{R}$ is their number. Therefore, $K_\mathbf{R}$ is the cardinality, $|\mathbf{R}|$, of $\mathbf{R}$.

Each event yields a unique system outcome, while different events may yield the same outcome. For instance, there might be more than one event that leads to system failure. In event trees, a complete joint event and its associated outcome are equivalent to a path [9, 10].

Let $T : \hat{e} \mapsto r = T(\hat{e})$, where $\hat{e} \in \mathbf{E}$ and $r \in \mathbf{R}$, denote the many-to-one mapping from the event space $\mathbf{E}$ to the set of system outcomes $\mathbf{R}$. The T mapping is recorded in the *system event table*, as a complete list of all the system events and their outcome.

The T defines a partition on $\mathbf{E}$, which is herein called the *outcome-based partition* and denoted by $\mathbf{Q}^T(\mathbf{E})$. The members of $\mathbf{Q}^T(\mathbf{E})$ are denoted by $\mathbf{q}_{r_i}^{\mathbf{Q}^T}(\mathbf{E})$, $r_i \in \mathbf{R}$.

Table 1 gives the event table of an example taken from Papazoglou [10]. In this case, there are 32 events and 4 possible outcomes, and $\mathbf{Q}^T(\mathbf{E})$ is comprised of the 4 sets:

(1) $\mathbf{q}_{r_1}^{\mathbf{Q}^T}(\mathbf{E}) = \{\widehat{e_1}, \widehat{e_3}, \widehat{e_4}, \widehat{e_{11}}, \widehat{e_{12}}, \widehat{e_{17}}, \widehat{e_{18}}, \widehat{e_{19}}, \widehat{e_{20}}, \widehat{e_{25}}, \widehat{e_{26}}, \widehat{e_{27}}, \widehat{e_{28}}\}$,

(2) $\mathbf{q}_{r_2}^{\mathbf{Q}^T}(\mathbf{E}) = \{\widehat{e_7}, \widehat{e_{15}}, \widehat{e_{16}}, \widehat{e_{21}}, \widehat{e_{22}}, \widehat{e_{23}}, \widehat{e_{24}}, \widehat{e_{29}}, \widehat{e_{30}}, \widehat{e_{31}}, \widehat{e_{32}}\}$,

(3) $\mathbf{q}_{r_3}^{\mathbf{Q}^T}(\mathbf{E}) = \{\widehat{e_5}, \widehat{e_6}, \widehat{e_8}, \widehat{e_{13}}, \widehat{e_{14}}\}$,

(4) $\mathbf{q}_{r_4}^{\mathbf{Q}^T}(\mathbf{E}) = \{\widehat{e_2}, \widehat{e_9}, \widehat{e_{10}}\}$.

Note that the T mapping can be derived from the structural dependencies among the system blocks, as dictated by the rational and physical interconnections of components within the system and depicted in the form of a fault tree or a functional block diagram. However, such information may be unavailable or too difficult to attain or process. This work assumes that the only information available is the system event table.

3. System Representation

The system representation presented here aims to organize the information contained in the event table. The table data are partitioned and organized into vectors summarizing

Table 1: Event table for the motor-operated valve [10].

Complete joint events	Block states			System outcomes
	b_1	b_2	b_3	
$\widehat{e_1}$	s_1^1	s_2^1	s_3^1	r_1
$\widehat{e_2}$	s_1^1	s_2^1	s_3^2	r_4
$\widehat{e_3}$	s_1^1	s_2^1	s_3^3	r_1
$\widehat{e_4}$	s_1^1	s_2^1	s_3^4	r_1
$\widehat{e_5}$	s_1^1	s_2^2	s_3^1	r_3
$\widehat{e_6}$	s_1^1	s_2^2	s_3^2	r_3
$\widehat{e_7}$	s_1^1	s_2^2	s_3^3	r_2
$\widehat{e_8}$	s_1^1	s_2^2	s_3^4	r_3
$\widehat{e_9}$	s_1^1	s_2^3	s_3^1	r_4
$\widehat{e_{10}}$	s_1^1	s_2^3	s_3^2	r_4
$\widehat{e_{11}}$	s_1^1	s_2^3	s_3^3	r_1
$\widehat{e_{12}}$	s_1^1	s_2^3	s_3^4	r_1
$\widehat{e_{13}}$	s_1^1	s_2^4	s_3^1	r_3
$\widehat{e_{14}}$	s_1^1	s_2^4	s_3^2	r_3
$\widehat{e_{15}}$	s_1^1	s_2^4	s_3^3	r_2
$\widehat{e_{16}}$	s_1^1	s_2^4	s_3^4	r_2
$\widehat{e_{17}}$	s_1^2	s_2^1	s_3^1	r_1
$\widehat{e_{18}}$	s_1^2	s_2^1	s_3^2	r_1
$\widehat{e_{19}}$	s_1^2	s_2^1	s_3^3	r_1
$\widehat{e_{20}}$	s_1^2	s_2^1	s_3^4	r_1
$\widehat{e_{21}}$	s_1^2	s_2^2	s_3^1	r_2
$\widehat{e_{22}}$	s_1^2	s_2^2	s_3^2	r_2
$\widehat{e_{23}}$	s_1^2	s_2^2	s_3^3	r_2
$\widehat{e_{24}}$	s_1^2	s_2^2	s_3^4	r_2
$\widehat{e_{25}}$	s_1^2	s_2^3	s_3^1	r_1
$\widehat{e_{26}}$	s_1^2	s_2^3	s_3^2	r_1
$\widehat{e_{27}}$	s_1^2	s_2^3	s_3^3	r_1
$\widehat{e_{28}}$	s_1^2	s_2^3	s_3^4	r_1
$\widehat{e_{29}}$	s_1^2	s_2^4	s_3^1	r_2
$\widehat{e_{30}}$	s_1^2	s_2^4	s_3^2	r_2
$\widehat{e_{31}}$	s_1^2	s_2^4	s_3^3	r_2
$\widehat{e_{32}}$	s_1^2	s_2^4	s_3^4	r_2

the contribution of block states in each subspace of the partition. These vectors will provide the framework to apply the manipulations described in Section 4.

3.1. Cartesian Subspaces and Partitions. Let $\mathbf{S}_k^{\mathbf{A}} \subseteq \mathbf{S}_k$, $\mathbf{A} \subseteq \mathbf{E} \wedge \mathbf{A} \neq \varnothing$ denote the set of b_k block states, $k \in \{1, 2, \ldots, K\}$, in the events comprising $\mathbf{A}$. Note that, since $\mathbf{A} \neq \varnothing$, then $\mathbf{S}_k^{\mathbf{A}} \neq \varnothing$, for all k.

Let $\mathbf{C}(\mathbf{A})$, $\mathbf{A} \subseteq \mathbf{E} \wedge \mathbf{A} \neq \varnothing$, denote the Cartesian product $\mathbf{S}_1^{\mathbf{A}} \times \mathbf{S}_2^{\mathbf{A}} \times \cdots \times \mathbf{S}_K^{\mathbf{A}}$. In general, $\mathbf{C}(\mathbf{A})$ is a superset of $\mathbf{A}$, since it always contains all the elements of $\mathbf{A}$ and may contain events not included in $\mathbf{A}$.

Consider the system of Table 1. For the subspace $\mathbf{B} = \{\widehat{e_1}, \widehat{e_3}, \widehat{e_4}, \widehat{e_{17}}, \widehat{e_{18}}, \widehat{e_{19}}, \widehat{e_{20}}\}$ we get $\mathbf{S}_1^{\mathbf{B}} = \{s_1^1, s_1^2\}$, $\mathbf{S}_2^{\mathbf{B}} = \{s_2^1\}$, $\mathbf{S}_3^{\mathbf{B}} = \{s_3^1, s_3^2, s_3^3, s_3^4\}$, and $\mathbf{C}(\mathbf{B}) = \{\widehat{e_1}, \widehat{e_2}, \widehat{e_3}, \widehat{e_4}, \widehat{e_{17}}, \widehat{e_{18}}, \widehat{e_{19}}, \widehat{e_{20}}\}$. As expected, $\mathbf{C}(\mathbf{B})$ is a superset of $\mathbf{B}$.

A *Cartesian subspace*, denoted as $\underline{\mathbf{A}}$, is herein defined as a nonempty subspace of $\mathbf{E}$, such that $\underline{\mathbf{A}} = \mathbf{C}(\underline{\mathbf{A}})$. Note that, all the singleton subspaces are Cartesian.

A *Cartesian partition* over $\mathbf{A}$, $\mathbf{A} \subseteq \mathbf{E} \wedge \mathbf{A} \neq \varnothing$, denoted as $\underset{\sim}{\mathbf{Q}}(\mathbf{A})$, is herein defined as a partition comprised only of Cartesian subspaces of $\mathbf{A}$. The elements of $\underset{\sim}{\mathbf{Q}}(\mathbf{A})$ are denoted as $\underset{\sim}{\mathbf{q}}_i^{\mathbf{Q}}(\mathbf{A})$, $i \in \{1, 2, \ldots, K_{\underset{\sim}{\mathbf{Q}}(\mathbf{A})}\}$. Note that, every subspace of $\mathbf{E}$ has at least one Cartesian partition comprised of singleton subspaces.

For instance, the subspace $\mathbf{B}$ defined above can be partitioned into the four subspaces $\underset{\sim}{\mathbf{q}}_1^{\mathbf{Q}}(\mathbf{B}) = \{\widehat{e_1}\}$, $\underset{\sim}{\mathbf{q}}_2^{\mathbf{Q}}(\mathbf{B}) = \{\widehat{e_3}\}$, $\underset{\sim}{\mathbf{q}}_3^{\mathbf{Q}}(\mathbf{B}) = \{\widehat{e_4}\}$, and $\underset{\sim}{\mathbf{q}}_4^{\mathbf{Q}}(\mathbf{B}) = \{\widehat{e_{17}}, \widehat{e_{18}}, \widehat{e_{19}}, \widehat{e_{20}}\}$. These are all Cartesian, since $\underset{\sim}{\mathbf{q}}_1^{\mathbf{Q}}(\mathbf{B})$, $\underset{\sim}{\mathbf{q}}_2^{\mathbf{Q}}(\mathbf{B})$, $\underset{\sim}{\mathbf{q}}_3^{\mathbf{Q}}(\mathbf{B})$ are singleton, and $\underset{\sim}{\mathbf{q}}_4^{\mathbf{Q}}(\mathbf{B}) = s_1^2 \times s_2^1 \times \mathbf{S}_3$.

3.2. Implicit Subspaces and Partitions. The state set $\mathbf{S}_k^{\mathbf{A}}$ is called a *complete state set* when it contains all the possible states of the b_k block, that is, $\mathbf{S}_k^{\mathbf{A}} = \mathbf{S}_k$.

An *implicit subspace*, denoted as $\underset{\approx}{\mathbf{A}}$, is herein defined as a Cartesian subspace when all of its constituent block state subsets $\mathbf{S}_k^{\underset{\approx}{\mathbf{A}}}$ are either complete or singleton sets. Note that an implicit subspace corresponds to the implicant [11] containing only the block states in the singleton sets. The subspace $\underset{\sim}{\mathbf{q}}_4^{\mathbf{Q}}(\mathbf{B})$ defined above is implicit and corresponds to the implicant $\{s_1^2, s_2^1\}$.

For example, the sets $\mathbf{B}_1 = s_1^2 \times s_2^2 \times \mathbf{S}_3$, $\mathbf{B}_2 = s_1^2 \times \{s_2^1, s_2^2\} \times \mathbf{S}_3$ and $\mathbf{B}_3 = s_1^2 \times \mathbf{S}_2 \times \mathbf{S}_3$ are all Cartesian (since they are Cartesian products) but $\mathbf{B}_2$ is not implicit.

An *implicit partition* over $\mathbf{A}$, denoted as $\underset{\approx}{\mathbf{Q}}(\mathbf{A})$, is herein defined as a partition comprised only of implicit subspaces. The elements of $\underset{\approx}{\mathbf{Q}}(\mathbf{A})$ are denoted as $\underset{\approx}{\mathbf{q}}_i^{\mathbf{Q}}(\mathbf{A})$. Every subspace $\mathbf{A}$ has at least oneimplicit partition, the $\underset{\approx}{\mathbf{Q}}(\mathbf{A}) = \mathbf{A}$, and the cardinalities of all possible partitions over $\mathbf{A}$ lie between one (when $\mathbf{A}$ is an implicit subspace) and $|\mathbf{A}|$.

For example, the partition over subspace $\mathbf{B}$ into $\underset{\sim}{\mathbf{q}}_i^{\mathbf{Q}}(\mathbf{B})$, $i \in \{1, 2, 3, 4\}$, is implicit. Also, the partition of $\mathbf{B}_2 = s_1^2 \times \{s_2^1, s_2^2\} \times \mathbf{S}_3$ into $s_1^2 \times s_2^1 \times \mathbf{S}_3$ and $s_1^2 \times s_2^2 \times \mathbf{S}_3$ is implicit.

An implicit partition conveys the same information as the set of its corresponding implicants. Therefore $\mathbf{B}_2$ can be derived from the prime implicants $\{s_1^2, s_2^1\}$ and $\{s_1^2, s_2^2\}$.

3.3. Contribution Vectors. The *contribution vector* of subspace $\mathbf{A}$, denoted by $\overline{\mathbf{N}}(\mathbf{A})$, is defined here as a vector of nonnegative integer entries $n_k^{i_k}(\mathbf{A}) \in \mathbb{Z}_+$, reporting the sum of the contributions of each state $s_k^{i_k}$ of each block b_k in the events comprising $\mathbf{A}$, namely, the number of times that each block state contributes in the development of $\mathbf{A}$.

For the example of subspace $\mathbf{B}$, the contribution vector is $\overline{\mathbf{N}}(\mathbf{B}) = [3,4;7,0,0,0;2,1,2,2]$, where the semicolons are used as separators between the three block compartments.

Properties of contribution vectors include the following.

(i) The vector length is $\lambda = \sum_{k=1}^{K} |\mathbf{S}_k|$, therefore $\overline{\mathbf{N}}(\mathbf{A}) \in \mathbb{N}_{+}^{\lambda}$, for all $\mathbf{A} \subseteq \mathbf{E}$.

(ii) For each block b_k, the sum of all the block entries in $\overline{\mathbf{N}}(\mathbf{A})$ equals $|\mathbf{A}|$.

In general, vector $\overline{\mathbf{N}}(\mathbf{A})$ is an abstract (inductive) representation of the information stored in $\mathbf{A}$. As the size of $\mathbf{A}$ increases, retrieving it from $\overline{\mathbf{N}}(\mathbf{A})$ is not trivial, it might even be impossible.

3.4. Bicontribution Vectors. Let the *bicontribution vector* $\overline{\mathbf{L}}(\mathbf{A})$ of subspace $\mathbf{A}$ be defined as a vector of Boolean entries $l_k^{i_k}(\mathbf{A}) \in \{\mathrm{O}, \mathrm{I}\}$, reporting the contribution or not of each state $s_k^{i_k}$ of each block b_k to the events of $\mathbf{A}$, namely, whether a certain block state contributes ("true" or I) or not ("false" or O) in the development of set $\mathbf{A}$.

Properties of bicontribution vectors include the following.

(i) The vector length is $\lambda = \sum_{k=1}^{K} |\mathbf{S}_k|$, so $\overline{\mathbf{L}}(\mathbf{A}) \in \{\mathrm{O},\mathrm{I}\}^{\lambda}$, for all $\mathbf{A} \subseteq \mathbf{E}$.

(ii) The sum of all the entries corresponding to block b_k equals $|\mathbf{S}_k^{\mathbf{A}}|$, for all $k \in \{1,2,\ldots,K\}$.

For instance, the associated bicontribution vector of $\overline{\mathbf{N}}(\mathbf{B})$ is $\overline{\mathbf{L}}(\mathbf{B}) = [\mathrm{I},\mathrm{I};\mathrm{I},\mathrm{O},\mathrm{O},\mathrm{I};\mathrm{I},\mathrm{I},\mathrm{I},\mathrm{I}]$. The sum of entries is 8 and equals the number of block state instances present in set $\mathbf{B}$.

Let $\Lambda[\overline{\mathbf{N}}(\mathbf{A})] : \{1,2,\ldots,|\mathbf{A}|\} \to \{\mathrm{I},\mathrm{O}\}$ denote the operation applied on the contribution vector $\overline{\mathbf{N}}(\mathbf{A})$ to derive its associated bicontribution vector $\overline{\mathbf{L}}(\mathbf{A})$. Based on the definitions of contribution and bicontribution vectors:

$$\overline{\mathbf{L}}(\mathbf{A}) = \Lambda\left[\overline{\mathbf{N}}(\mathbf{A})\right] \Longleftrightarrow l_k^{i_k}(\mathbf{A}) = \begin{cases} \mathrm{O}, & n_k^{i_k}(\mathbf{A}) = 0, \\ \mathrm{I}, & n_k^{i_k}(\mathbf{A}) > 0, \end{cases} \quad \forall k, i_k. \tag{1}$$

The reverse operation $\Lambda^{-1}[\overline{\mathbf{N}}(\mathbf{A})]$ yields $\mathbf{C}(\mathbf{A})$ as follows:

$$\begin{aligned}\Lambda^{-1}&\left[\overline{\mathbf{N}}(\mathbf{A})\right] \\ &= \left\{\hat{e} = \left\{s_1^{i_1}, s_2^{i_2}, \ldots, s_K^{i_K}\right\} \in \mathbf{E} : l_k^{i_k}(\mathbf{A}) = \mathrm{I}, \forall k, i_k\right\}.\end{aligned} \tag{2}$$

Therefore, from the definition of $\mathbf{C}(\mathbf{A})$, $\Lambda^{-1}[\overline{\mathbf{L}}(\mathbf{A})] = \mathbf{C}(\mathbf{A})$. Clearly, $\overline{\mathbf{L}}(\mathbf{A})$ always carries less information than its associated vector $\overline{\mathbf{N}}(\mathbf{A})$, unless $\mathbf{A}$ is Cartesian.

3.5. Cartesian Contribution Vectors. The *Cartesian contribution vector* of subspace $\mathbf{A}$, denoted as $\underset{\sim}{\overline{\mathbf{N}}}(\mathbf{A})$, is defined here as the contribution vector of $\mathbf{C}(\mathbf{A})$, thus $\underset{\sim}{\overline{\mathbf{N}}}(\mathbf{A}) \equiv \overline{\mathbf{N}}(\mathbf{C}(\mathbf{A}))$. Let $\underset{\sim}{n}_k^{i_k}(\mathbf{A})$ denote the entries of $\underset{\sim}{\overline{\mathbf{N}}}(\mathbf{A})$.

For instance, starting from subspace $\mathbf{B}$ we get $\underset{\sim}{\overline{\mathbf{N}}}(\mathbf{B}) = [4,4;8,0,0,0;2,2,2,2]$.

Let $\Xi[\overline{\mathbf{L}}(\mathbf{A})]$ denote the operation applied on the bicontribution vector $\overline{\mathbf{L}}(\mathbf{A})$ to derive the Cartesian contribution vector $\underset{\sim}{\overline{\mathbf{N}}}(\mathbf{A})$. Based on the above vector definitions:

$$\underset{\sim}{\overline{\mathbf{N}}}(\mathbf{A}) = \Xi\left[\overline{\mathbf{L}}(\mathbf{A})\right] \Longleftrightarrow \underset{\sim}{n}_k^{i_k}(\mathbf{A}) = \begin{cases} 0, & l_k^{i_k}(\mathbf{A}) = \mathrm{O}, \\ \dfrac{|\mathbf{C}(\mathbf{A})|}{\left|\mathbf{S}_k^{\mathbf{A}}\right|}, & l_k^{i_k}(\mathbf{A}) = \mathrm{I}, \end{cases} \quad \forall k, i_k. \tag{3}$$

For every subspace $\mathbf{A}$, each entry of $\mathbf{C}(\mathbf{A})$, $n_k^{i_k}(\mathbf{A})$, lays between zero and $\underset{\sim}{n}_k^{i_k}(\mathbf{A})$. Nonzero entries for which $n_k^{i_k}(\mathbf{A}) = \underset{\sim}{n}_k^{i_k}(\mathbf{A})$ are herein called *Cartesian entries*.

The vector $\overline{\mathbf{N}}(\mathbf{A})$ satisfies the *Cartesian property* if and only if is $\mathbf{A}$ a Cartesian subset. In this case $\overline{\mathbf{N}}(\mathbf{A}) = \underset{\sim}{\overline{\mathbf{N}}}(\mathbf{A})$.

Going back to set $\mathbf{B}$ and comparing $\overline{\mathbf{N}}(\mathbf{B})$ to $\underset{\sim}{\overline{\mathbf{N}}}(\mathbf{B})$:

(i) the entries corresponding to the block states s_1^4, s_4^1, s_4^3, and s_4^4 are Cartesian,

(ii) the $\overline{\mathbf{N}}(\mathbf{B})$ vector does not satisfy the Cartesian property since $n_k^{i_k}(\mathbf{B}) \neq \underset{\sim}{n}_k^{i_k}(\mathbf{B})$ at $\{k, i_k\} = \{2,1\}$ or $\{3,2\}$.

The contribution vectors of implicit subspaces are herein called *implicit contribution vectors*. Since, implicit subspaces are always Cartesian, implicit contribution vectors always satisfy the Cartesian property.

3.6. Vector Partitions. Considering a partition $\mathbf{Q}(\mathbf{A})$, the *contribution vector partition* of $\mathbf{Q}(\mathbf{A})$, denoted as $\overline{\overline{\mathbf{N}}}(\mathbf{Q}(\mathbf{A}))$, is defined as the set of the contribution vectors $\overline{\mathbf{N}}(\mathbf{q}_i^{\mathbf{Q}}(\mathbf{A}))$.

Since $\sum_i n_k^{i_k}(\mathbf{q}_i^{\mathrm{Q}}(\mathbf{A})) = n_k^{i_k}(\mathbf{A})$, for all k, i_k, the vector $\overline{\mathbf{N}}(\mathbf{A})$ is specified as the *composite contribution vector* of $\overline{\overline{\mathbf{N}}}(\mathbf{Q}(\mathbf{A}))$. In general, composite contribution vectors carry less information than their vector partitions.

Likewise to subspace partitions, vector partitions can be classified as Cartesian (when all their associated vectors are Cartesian) or implicit (when all their associated vectors are implicit). Cartesian and implicit partitions are denoted as $\underset{\sim}{\overline{\overline{\mathbf{N}}}}(\bullet)$ and $\underset{\approx}{\overline{\overline{\mathbf{N}}}}(\bullet)$, respectively.

Considering subspace $\mathbf{B}$ and its partition $\underset{\sim}{\mathbf{q}}_1^{\mathbf{Q}}(\mathbf{B}) = \{\widehat{e}_1\}$, $\underset{\sim}{\mathbf{q}}_2^{\mathbf{Q}}(\mathbf{B}) = \{\widehat{e}_3\}$, $\underset{\sim}{\mathbf{q}}_3^{\mathbf{Q}}(\mathbf{B}) = \{\widehat{e}_4\}$, and $\underset{\sim}{\mathbf{q}}_4^{\mathbf{Q}}(\mathbf{B}) = \{\widehat{e}_{17}, \widehat{e}_{18}, \widehat{e}_{19}, \widehat{e}_{20}\}$, the respective contribution vector partition is:

$$\overline{\overline{\mathbf{N}}}(\mathbf{Q}(\mathbf{B})) = \begin{Bmatrix} \overline{\mathbf{N}}\left(\underset{\sim}{\mathbf{q}}_1^{\mathbf{Q}}(\mathbf{B})\right) \\ \overline{\mathbf{N}}\left(\underset{\sim}{\mathbf{q}}_2^{\mathbf{Q}}(\mathbf{B})\right) \\ \overline{\mathbf{N}}\left(\underset{\sim}{\mathbf{q}}_3^{\mathbf{Q}}(\mathbf{B})\right) \\ \overline{\mathbf{N}}\left(\underset{\sim}{\mathbf{q}}_4^{\mathbf{Q}}(\mathbf{B})\right) \end{Bmatrix} = \begin{Bmatrix} [1,0;1,0,0,0;1,0,0,0] \\ [1,0;1,0,0,0;0,0,1,0] \\ [1,0;1,0,0,0;0,0,0,1] \\ [0,4;4,0,0,0;1,1,1,1] \end{Bmatrix}. \tag{4}$$

The $\underset{\sim}{\mathbf{q}}_i^{\mathbf{Q}}(\mathbf{B})$ subsets are implicit, so $\overline{\overline{\mathbf{N}}}(\mathbf{Q}(\mathbf{B}))$ is an implicit contribution vector partition.

It should be highlighted that a Cartesian contribution vector partition is a disjoint-set data structure, as it contains all the information necessary to: (a) find which subspace includes a certain event and (b) reconstruct the event union set using the $\Lambda^{-1}[\overline{\mathbf{L}}(\bullet)]$ operations.

4. Decomposition and Recomposition Operations

Stating from a given outcome-based partition $\mathbf{Q}^T(\mathbf{E})$, the scope here is to derive an implicit partition featuring minimal cardinality for each of the $\mathbf{q}_{r_i}^{\mathbf{Q}^T}(\mathbf{E})$, $r_i \in \mathbf{R}$ subspaces. This would be equivalent to a set of prime implicants describing the event table. The scope is accomplished through the application of specific operations on the system vectors defined above. These operations extract knowledge from the information carried in the composition vectors and store this knowledge in the minimal possible schemes. The naming of these operations is after Shannon's decomposition [11].

4.1. Decomposition of Contribution Vectors. This section presents an operation for the systematic manipulation of a contribution vector to yield a contribution vector partition. The operation is called *decomposition*, since it is a case of multistate Shannon's decomposition applied on the contribution vectors. This operation decreases the abstraction of the information stored in the vector, since contribution vector partitions carry more information than their composite vectors. Starting from a contribution vector, subsequent decomposition actions generate a low cardinality implicit partition of this vector.

Consider a subspace $\mathbf{A} \neq \varnothing : \exists k : |\mathbf{S}_k^{\mathbf{A}}| \geq 2$ and a nonempty subset $\sigma_k \subset \mathbf{S}_k^{\mathbf{A}}$. The *decomposition operation* of $\mathbf{A}$ according to σ_k is a partition rule applied on $\mathbf{A}$ to generate the sets $\{\mathbf{A}^{\sigma_k}, \mathbf{A} - \mathbf{A}^{\sigma_k}\}$ such that:

$$\mathbf{A}^{\sigma_k} = \left\{\hat{e} \in \mathbf{A} \cap \Delta_{\mathbf{A}}^{\sigma_k}\right\},$$
$$\mathbf{A} - \mathbf{A}^{\sigma_k} = \left\{\hat{e} \in \mathbf{A} \cap \left(\mathbf{C}(\mathbf{A}) - \Delta_{\mathbf{A}}^{\sigma_k}\right)\right\}, \quad (5)$$
$$\text{where } \Delta_{\mathbf{A}}^{\sigma_k} = \left\{\mathbf{S}_1^{\mathbf{A}} \times \mathbf{S}_2^{\mathbf{A}} \times \cdots \times \sigma_k \times \cdots \times \mathbf{S}_K^{\mathbf{A}}\right\}.$$

Since $\sigma_k \neq \varnothing$ then $\mathbf{A}^{\sigma_k}, \mathbf{A} - \mathbf{A}^{\sigma_k} \neq \varnothing$.

The simplest way of applying the decomposition operation is to go through each one of the events in $\mathbf{A}$, as described in the operation definition. This procedure requires computational effort to decide whether a certain event belongs in $\mathbf{A}^{\sigma_k}$ or $\mathbf{A} - \mathbf{A}^{\sigma_k}$. If, on the other hand, $\overline{\mathbf{N}}(\mathbf{A})$ contains Cartesian entries, the operation can be applied directly on $\overline{\mathbf{N}}(\mathbf{A})$. Section 6 shows that decomposing $\overline{\mathbf{N}}(\mathbf{A})$ rather than $\mathbf{A}$ reduces the computational effort by several orders of magnitude.

Consider a contribution vector $\overline{\mathbf{N}}(\mathbf{A}) : \exists k : |\mathbf{S}_k^{\mathbf{A}}| \geq 2 \wedge |\mathbf{S}_k^{\mathbf{A}}| \neq 0$ and a nonempty set $\sigma_k \subseteq \mathbf{S}_k^{\mathbf{A}}$. The *vector decomposition operation* according to σ_k is a partition rule applied on $\overline{\mathbf{N}}(\mathbf{A})$ to yield the contribution vector partition $\{\overline{\mathbf{N}}(\mathbf{A}^{\sigma_k}), \overline{\mathbf{N}}(\mathbf{A}^{\overline{\sigma}_k})\}$ such that

$$\overline{\mathbf{N}}(\mathbf{A}^{\sigma_k}) = \overline{\mathbf{N}}\left(\mathbf{S}_1^{\mathbf{N}} \times \mathbf{S}_2^{\mathbf{N}} \times \cdots \times \sigma_k \times \cdots \times \mathbf{S}_K^{\mathbf{N}}\right),$$
$$\overline{\mathbf{N}}(\mathbf{A}^{\overline{\sigma}_k}) = \overline{\mathbf{N}}(\mathbf{A}) - \overline{\mathbf{N}}(\mathbf{A}^{\sigma_k}). \quad (6)$$

The development of $\overline{\mathbf{N}}(\mathbf{A}^{\sigma_k})$ ensures that this vector has the Cartesian property. Instead of the Cartesian product $\mathbf{S}_1^{\mathbf{N}} \times \mathbf{S}_2^{\mathbf{N}} \times \cdots \times \sigma_k \times \cdots \times \mathbf{S}_K^{\mathbf{N}}$ we can use the bicontribution vector:

$$\overline{\mathbf{L}}(\mathbf{A}^{\sigma_k}) : l_m^{i_m}\big|_{\overline{\mathbf{L}}(\mathbf{A}^{\sigma_k})} = \begin{cases} l_m^{i_m}\big|_{\overline{\mathbf{L}}(\mathbf{A})}, & k \neq m, \\ \mathrm{I}, & k = m \wedge s_m^{i_m} \in \sigma_k, \\ \mathrm{O}, & k = m \wedge s_m^{i_{m,n}} \notin \sigma_k. \end{cases} \quad (7)$$

The decomposition operation can be applied iteratively. To simplify the notation, let $\overline{\mathbf{N}}_0$ be an initial contribution vector, decomposed into $\{\overline{\mathbf{N}}_1^{\sigma_0}, \overline{\mathbf{N}}_1\}$, where $\overline{\mathbf{N}}_1^{\sigma_0}$ has the Cartesian property. Then, $\overline{\mathbf{N}}_1$ is decomposed into $\{\overline{\mathbf{N}}_2^{\sigma_1}, \overline{\mathbf{N}}_2\}$, where $\overline{\mathbf{N}}_2^{\sigma_1}$ has the Cartesian property and so forth. Application of the decomposition operation over i iterations replaces $\overline{\mathbf{N}}_0$ with the contribution vector partition $\overline{\overline{\mathbf{N}}}_i = \{\overline{\mathbf{N}}_1^{\sigma_0}, \overline{\mathbf{N}}_2^{\sigma_1}, \overline{\mathbf{N}}_3^{\sigma_2}, \ldots, \overline{\mathbf{N}}_i^{\sigma_{i-1}}, \overline{\mathbf{N}}_i\}$. The iterations terminate when $\overline{\mathbf{N}}_i$ has the Cartesian property, therefore $\overline{\overline{\mathbf{N}}}_i$ is a Cartesian partition. Note that, the $\overline{\mathbf{N}}_i^{\sigma_{i-1}}$ vectors can be derived as $\overline{\mathbf{N}}_i^{\sigma_{i-1}} = \Xi[\overline{\mathbf{L}}_i^{\sigma_{i-1}}]$, using (7) and $\overline{\mathbf{L}}_{i-1} = \Lambda[\overline{\mathbf{N}}_{i-1}]$.

4.1.1. Decomposition Example. To illustrate the decomposition operation consider the set $\mathbf{q}_{r_1}^T(\mathbf{E})$ of Table 1 and its contribution vector $\overline{\mathbf{N}}_0 = [5, 8; 7, 0, 6, 0; 3, 2, 4, 4]$.

Starting from $\overline{\mathbf{N}}_0$, the bicontribution vector is $\overline{\mathbf{L}}_0 = [\mathrm{I}, \mathrm{I}; \mathrm{I}, \mathrm{O}, \mathrm{I}, \mathrm{O}; \mathrm{I}, \mathrm{I}, \mathrm{I}, \mathrm{I}]$ and the Cartesian contribution vector is $\overline{\mathbf{N}}_{\sim 0} = [8, 8; 8, 0, 8, 0; 4, 4, 4, 4]$. Clearly, $\overline{\mathbf{N}}_{\sim 0}$ has three Cartesian entries at s_1^2, s_3^3 and s_3^4. Let $\sigma_0 = \{s_3^3\}$:

(i) $\overline{\mathbf{L}}_1^{\sigma_0} = [\mathrm{I}, \mathrm{I}; \mathrm{I}, \mathrm{O}, \mathrm{I}, \mathrm{O}; \mathrm{O}, \mathrm{O}, \mathbf{I}, \mathrm{O}]$

$\Rightarrow \overline{\mathbf{N}}_1^{\sigma_0} = [2, 2; 2, 0, 2, 0; 0, 0, \mathbf{4}, 0]$,

(ii) $\overline{\mathbf{N}}_1 = \overline{\mathbf{N}}_0 - \overline{\mathbf{N}}_1^{\sigma_0} = [3, 6; 5, 0, 4, 0; 3, 2, \mathbf{0}, 4]$,

$\overline{\mathbf{L}}_1 = [\mathrm{I}, \mathrm{I}; \mathrm{I}, \mathrm{O}, \mathrm{I}, \mathrm{O}; \mathrm{I}, \mathrm{I}, \mathbf{O}, \mathrm{I}]$.

Alternatively, the decomposition could be applied *simultaneously* at $\sigma_0^* = \{s_3^3, s_3^4\}$ as follows:

(i) $\overline{\mathbf{L}}_1^{\sigma_0^*} = [\mathrm{I}, \mathrm{I}; \mathrm{I}, \mathrm{O}, \mathrm{I}, \mathrm{O}; \mathrm{O}, \mathrm{O}, \mathbf{I}, \mathbf{I}]$

$\Rightarrow \overline{\mathbf{N}}_1^{\sigma_0^*} = [4, 4; 4, 0, 4, 0; 0, 0, \mathbf{4}, \mathbf{4}]$,

(ii) $\overline{\mathbf{N}}_1^* = \overline{\mathbf{N}}_0 - \overline{\mathbf{N}}_1^{\sigma_0^*} = [1, 4; 3, 0, 2, 0; 3, 2, \mathbf{0}, \mathbf{0}]$,

$\overline{\mathbf{L}}_1^* = [\mathrm{I}, \mathrm{I}; \mathrm{I}, \mathrm{O}, \mathrm{I}, \mathrm{O}; \mathrm{I}, \mathrm{I}, \mathbf{O}, \mathbf{O}]$.

In both cases, the vectors $\overline{\mathbf{N}}_1$ and $\overline{\mathbf{N}}_1^*$ do not satisfy the Cartesian property, but they contain Cartesian entries. For instance, $\overline{\mathbf{N}}_1^*$ can be decomposed at $\sigma_1^* = \{s_1^2\}$ as follows:

(i) $\overline{\mathbf{L}}_2^{\sigma_1^*} = [0, \mathbf{1}; 1, 0, 1, 0; 1, 1, 0, 0]$

$\Rightarrow \overline{\mathbf{N}}_2^{\sigma_1^*} = [0, \mathbf{4}; 2, 0, 2, 0; 2, 2, 0, 0]$,

(ii) $\overline{\mathbf{N}}_2^* = \overline{\mathbf{N}}_1^* - \overline{\mathbf{N}}_2^{\sigma_1^*} = [1, \mathbf{0}; 1, 0, 0, 0; 1, 0, 0, 0]$.

Both $\overline{\mathbf{N}}_2^*$ and $\overline{\mathbf{N}}_2^{\sigma_1^*}$ satisfy the Cartesian property and no further decomposition is necessary. So, the decomposition of $\overline{\mathbf{N}}_0$ yields the Cartesian contribution vector partition $\{\overline{\mathbf{N}}_1^{\sigma_0}, \overline{\mathbf{N}}_2^*, \overline{\mathbf{N}}_2^{\sigma_1^*}\}$.

The set vectors may contain complete blocks, like the first block in $\overline{\mathbf{N}}_1^{\sigma_0}$. The vectors $\overline{\mathbf{N}}_1^{\sigma_0}$ and $\overline{\mathbf{N}}_2^{\sigma_1}$ can be further decomposed until they give implicit contribution vector partitions:

(i)

$$\overline{\mathbf{N}}_1^{\sigma_0} = [4, 4; 4, 0, 4, 0; 0, 0, 4, 4] \sim \left\{ \frac{[2, 2; 4, 0, 0, 0; 0, 0, 2, 2]}{[2, 2; 0, 0, 4, 0; 0, 0, 2, 2]} \right\} \sim \left\{ \begin{matrix} [1, 1; 2, 0, 0, 0; 0, 0, 2, 0] \\ [1, 1; 2, 0, 0, 0; 0, 0, 0, 2] \\ \hline [1, 1; 0, 0, 2, 0; 0, 0, 2, 0] \\ [1, 1; 0, 0, 2, 0; 0, 0, 0, 2] \end{matrix} \right\}, \tag{8}$$

(ii)

$$\overline{\mathbf{N}}_2^{\sigma_1} = [0, 4; 2, 0, 2, 0; 2, 2, 0, 0] \sim \left\{ \frac{[0, 2; 0, 0, 2, 0; 1, 1, 0, 0]}{[0, 2; 2, 0, 0, 0; 1, 1, 0, 0]} \right\} \sim \left\{ \begin{matrix} [0, 1; 0, 0, 1, 0; 0, 1, 0, 0] \\ [0, 1; 0, 0, 1, 0; 1, 0, 0, 0] \\ \hline [0, 1; 1, 0, 0, 0; 0, 1, 0, 0] \\ [0, 1; 1, 0, 0, 0; 1, 0, 0, 0] \end{matrix} \right\}. \tag{9}$$

In the previous example, the cardinality of $\mathbf{q}_{r_1}^T(\mathbf{E})$ was 13, while the cardinality of the implicit partition is equal to $4+4+1 = 9$. The latter cardinality could be even smaller if a more intelligent decomposition strategy were applied. For instance, $\overline{\mathbf{N}}_0$ has two complete blocks (first and third block) and three Cartesian entries (one in the first block and two in the third). The choice of decomposition order, that is, the sequence of σ_i's is crucial, as discussed in the algorithm implementation section.

4.2. Recomposition of Contribution Vectors. The outcome of the decomposition operation is an implicit contribution vector partition. Given this, we can seek merging opportunities to create unions that have complete blocks, thus reduce the partition cardinality. Since the contribution vectors are all implicit, the recomposition operation is hereby discussed in terms of bicontribution vectors.

Consider a block b_m and its set of states $\mathbf{S}_m$. Let $\overline{\overline{\mathbf{L}}}$ be a set of $|\mathbf{S}_m|$ bicontribution vectors, $\overline{\mathbf{L}}_i = [l_{i_m,k}^{i_k}]$, $i_m \in \{1, 2, \ldots, |\mathbf{S}_m|\}$, $i_k \in \{1, 2, \ldots, |\mathbf{S}_k|\}$, $k \in \{1, 2, \ldots, K\}$, such that $l_{i_m,k}^{i_k} = \lambda_k^{i_k} \in \{\mathrm{O}, \mathrm{I}\}$, for all $k \neq m$, i_m, i_k. Then, the set $\overline{\overline{\mathbf{L}}}$ can be replaced by the single bicontribution vector $\overline{\mathbf{L}} = [\lambda_k^{i_k}]$, where $\lambda_m^{i_m} = I$, for all i_m. Note that, $\overline{\mathbf{L}} = \overline{\mathbf{L}}(\mathbf{S}_1^{\overline{\mathbf{L}}_1} \times \mathbf{S}_2^{\overline{\mathbf{L}}_1} \times \cdots \times \mathbf{S}_m \times \cdots \times \mathbf{S}_K^{\overline{\mathbf{L}}_1})$.

4.2.1. Recomposition Example. Consider the set $\overline{\overline{\mathbf{L}}} = \left\{ \begin{matrix} [\mathrm{I, O; I, O, O, O; I, O, O, O}] \\ [\mathrm{O, I; I, O, O, O; I, O, O, O}] \end{matrix} \right\}$.

From the decomposition example discussed earlier. The two bicomposition vectors have exactly the same entries in the second and third block but different entries in the first block. In addition, the first block has exactly two state instances. The two vectors can be recomposed into a single equivalent vector [I, I; I, O, O, O; I, O, O, O].

5. Algorithm Implementation

Given the table associated with an event tree, the proposed algorithm launches an iterative process of decomposition and recomposition operations until we obtain an implicit partition of minimal cardinality for each one of the tree outcomes. Working with the vectors defined above rather than sets of events decreases significantly the amount of information being stored and the computational effort for manipulating this information.

The algorithm applies the vector decomposition and recomposition operations using a set of heuristic rules. These rules help in the identification of more promising entries to apply the operations. Sections 5.1 and 5.2 describe the heuristic rules, and Section 5.3 discusses how these procedures work together within the proposed algorithm.

5.1. Heuristic Rules for Decomposition Order. The decomposition operations proceed iteratively, replacing the original $|\mathbf{R}|$ vectors of the outcome partition with $|\mathbf{R}|$ contribution vector sets. Decompositions are applied locally, based on the features (e.g., the Cartesian entries) of each contribution vector.

Given a composition vector, the choice of decomposition order is crucial in preserving the maximum possible of the initial complete blocks. The set $\mathbf{q}_{r_1}^T(\mathbf{E})$ of Table 1 has a complete block of 2 states and a complete block of 4 states. If the decomposition order is different the final implicit partition could be different. In effect, applying the decomposition on entries $\{s_3^3, s_3^4\}$ of $\overline{\mathbf{N}}_0$ results in a reduction by $4 = 13 - 9$ on the cardinality of the final implicit partition of $\mathbf{q}_{r_1}^T(\mathbf{E})$. Applying the decomposition on $\{s_1^2\}$—followed by $\{s_3^3, s_3^4\}$, $\{s_3^1\}$, $\{s_4^3, s_4^4\}$, and so forth, in any order—preserves the third block, and the final reduction is by 13 − 7 = 6:

$$\begin{aligned}\overline{\mathbf{N}}_0 &= [5,8;7,0,6,0;3,2,4,4] \sim \left\{\begin{array}{c}[0,8;4,0,4,0;2,2,2,2]\\ \hline [5,0;3,0,2,0;1,0,2,2]\end{array}\right\}\\ &\sim \left\{\begin{array}{c}[0,4;4,0,0,0;1,1,1,1]\\ [0,4;0,0,4,0;1,1,1,1]\\ \hline [4,0;2,0,2,0;0,0,2,2]\\ [1,0;1,0,0,0;1,0,0,0]\end{array}\right\}\\ &\sim \left\{\begin{array}{cc}\begin{array}{c}[0,4;4,0,0,0;1,1,1,1]\\ [0,4;0,0,4,0;1,1,1,1]\\ \hline [1,0;1,0,0,0;1,0,0,0]\end{array} & \begin{array}{c}[1,0;1,0,0,0;0,0,1,0]\\ [1,0;1,0,0,0;0,0,0,1]\\ [1,0;0,0,1,0;0,0,1,0]\\ [1,0;0,0,1,0;0,0,0,1]\end{array}\end{array}\right\}.\end{aligned} \tag{10}$$

The following decomposition rules support the generation of the smallest possible implicit partitions at the minimum possible execution time, and they are applied on each contribution vector that is not Cartesian.

(a) If there are Cartesian entries in the current contribution vector:

  (i) it is prefered to decompose at Cartesian entries in incomplete blocks, to avoid breaking the complete blocks. Note that, in this case, the decomposition order makes no difference to the final partitions,

  (ii) if the only Cartesian entries are within complete blocks, start decomposing the complete blocks with the largest span between their contribution values, relatively to their number of states, that is, sort blocks according to $(\max_k\{n_k^{i_k}\} - \min_k\{n_k^{i_k}\})/|\mathbf{S}_k|$.

(b) If there are no Cartesian entries in the current contribution vector, then:

  (i) if no complete blocks are present, prefer incomplete blocks with more states being present, and decompose them into as many vectors as the number of states present,

  (ii) if only complete blocks are present, decompose the block featuring the entry with the maximum departure from its Cartesian value, that is, $\max_k\{\underset{\sim}{n}_k^{i_k} - n_k^{i_k}\}$.

Before the application of these rules it is essential to recognize which of the complete blocks make sense, according to the values of the contribution vector entries. For instance, the vector $[5,8;7,0,6,0;3,2,4,4]$ could lead to a partition that includes a bicontribution vector of 2 complete blocks, but this not possible for the vector $[3,8;5,0,6,0;2,2,4,3]$, since the first entry in the compartment of the first block is lower than the number of states in the third block. Similarly, in the vector $[0,4;2,0,2,0;1,1,1,1]$ the third block cannot remain complete under any decomposition order since there is no single entry in the second block compartment greater or equal to $|\mathbf{S}_3| = 4$.

The final Cartesian partition includes vectors and may need to be further decomposed to derive implicit partitions. In this case, the vectors are decomposed in all their incomplete block entries (which are all Cartesian), and the decomposition order makes no difference.

5.2. Heuristic Rules for Recomposition Order. Once the decomposition stage is completed, the final implicit partitions may have vectors that can be merged. This reduces the partition cardinalities. The choice of recomposition order is crucial, since the application of decomposition operations on large event tables can produce numerous vectors as candidates for recomposition.

Let $\mathbf{Q}_0(\mathbf{q}_{r_i}^T(\mathbf{E}))$, $r_i \in \mathbf{R}$, denote the implicit partitions output from the decomposition stage. The recomposition algorithm proceeds according to the following steps applied on each vector partition $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_0 = \overline{\overline{\underset{\sim}{\mathbf{N}}}}(\mathbf{Q}_0(\mathbf{q}_{r_i}^T(\mathbf{E})))$ (or the associated bicontribution vectors).

(i) Partition $\overline{\overline{\underset{\sim}{\mathbf{N}}}}$ into sets having the same complete blocks. Let $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_a$ be such a subset of $\overline{\overline{\underset{\sim}{\mathbf{N}}}}$.

(ii) The set $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_a$ is a candidate for creating an additional complete block b_m, if there is at least one subset $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_a^{m_i} \subseteq \overline{\overline{\underset{\sim}{\mathbf{N}}}}_a$ of cardinality $|\mathbf{S}_m|$, where $i \in I_m$ is the index of the particular subset for b_m, such that: (i) its overall contribution vector features b_m as a complete block and (ii) the remaining blocks have the same state sets $\mathbf{S}_k^{m_i}$, for all $k \neq m$.

(iii) If $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_a^{m_i}$ is replaced by its composite contribution vector, the resulting vector partition $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_1 = \overline{\overline{\underset{\sim}{\mathbf{N}}}}(\mathbf{Q}_1(\mathbf{q}_{r_i}^T(\mathbf{E})))$ is implicit.

(iv) The process is repeated on the new set $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_1$ to reduce it to $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_2$ and so forth and terminates when there are no more recomposition candidates left.

The set $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_a$ may have several subsets $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_a^{m_i}$ that relate to the creation of different complete blocks and/or to different ways of creating a particular complete block. The recomposition procedure is supported by intelligent selection biases to ensure that the merging opportunities are properly exploited. During the iterative recomposition process, the following recomposition rules are applied on every set $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_a$ of Cartesian contribution vectors featuring the same complete blocks.

(i) Find all the candidate sets $\overline{\overline{\underset{\sim}{\mathbf{N}}}}_a^{m_i} \subseteq \overline{\overline{\underset{\sim}{\mathbf{N}}}}_a$, $i \in I_m$, for creating each new complete block at b_m, for all $m \in \{1,2,\ldots,K\}$.

(ii) Associate each candidate to a score $w_a^{m_i}$ proportional to $|\mathbf{S}_m|$ and I_m and incorporate the potential to create two complete blocks in one go.

(iii) Select among conflicting sets according to their $w_a^{m_i}$.

Table 2: Recomposition example: initial set of bicontribution vectors (Step 1).

$\overline{\mathbf{L}}_i$	b_1				b_2		b_3			b_4		b_5		b_6		b_7		b_8		b_9		b_{10}	
24	O	O	O	I	I	O	O	I	O	O	I	O	I	I	O	I	O	I	O	**I**	**O**	I	O
26	O	O	O	I	I	O	O	I	O	O	I	O	I	I	O	I	O	I	O	**O**	**I**	I	O
28	O	O	O	I	I	O	O	I	O	O	I	O	I	O	I	I	O	I	O	I	I	I	O
21	I	O	O	O	I	O	I	I	I	O	I	O	I	I	O	I	O	I	O	I	O	I	O
25	I	O	O	O	I	O	I	I	I	O	I	O	I	I	O	I	O	I	O	O	I	I	O
11	O	O	O	I	I	O	O	I	O	I	O	I	O	I	I	I	O	I	O	I	I	I	O
12	O	O	O	I	I	O	O	I	O	O	I	I	O	I	I	I	O	I	O	I	I	I	O
18	O	O	O	I	I	O	O	I	O	I	O	O	I	I	I	I	O	I	O	I	I	I	O
22	O	O	O	I	I	O	I	O	O	O	I	O	I	I	O	I	I	I	I	I	O	I	O
23	O	O	O	I	I	O	O	O	I	O	I	O	I	I	O	I	I	I	I	I	O	I	O
27	I	O	O	O	I	O	I	I	I	O	I	O	I	O	I	I	O	I	O	I	I	I	O
5	I	O	O	O	I	O	I	I	I	I	O	I	O	I	I	I	O	I	O	I	I	I	O
6	I	O	O	O	I	O	I	I	I	O	I	I	O	I	I	I	O	I	O	I	I	I	O
14	I	O	O	O	I	O	I	I	I	I	O	O	I	I	I	I	O	I	O	I	I	I	O
15	O	O	I	O	I	O	I	I	I	I	O	O	I	I	O	I	I	I	I	I	O	I	O
19	O	I	O	O	I	O	I	I	I	O	I	O	I	I	O	I	I	I	I	I	O	I	O
20	O	O	I	O	I	O	I	I	I	O	I	O	I	I	O	I	I	I	I	I	O	I	O
7	O	O	O	I	I	O	I	O	O	I	O	I	O	I	I	I	I	I	I	I	I	I	O
8	O	O	O	I	I	O	I	O	O	O	I	I	O	I	I	I	I	I	I	I	I	I	O
9	O	O	O	I	I	O	O	O	I	I	O	I	O	I	I	I	I	I	I	I	I	I	O
10	O	O	O	I	I	O	O	O	I	O	I	I	O	I	I	I	I	I	I	I	I	I	O
16	O	O	O	I	I	O	I	O	O	I	O	O	I	I	I	I	I	I	I	I	I	I	O
17	O	O	O	I	I	O	O	O	I	I	O	O	I	I	I	I	I	I	I	I	I	I	O
29	O	O	O	I	I	O	I	O	O	I	O	I	O	I	I	I	I	I	I	I	I	O	I
30	O	O	O	I	I	O	I	O	O	I	O	O	I	I	I	I	I	I	I	I	I	O	I
1	O	I	O	O	I	O	I	I	I	I	O	I	O	I	I	I	I	I	I	I	I	I	O
2	O	I	O	O	I	O	I	I	I	O	I	I	O	I	I	I	I	I	I	I	I	I	O
3	O	O	I	O	I	O	I	I	I	I	O	I	O	I	I	I	I	I	I	I	I	I	O
4	O	O	I	O	I	O	I	I	I	O	I	I	O	I	I	I	I	I	I	I	I	I	O
13	O	I	O	O	I	O	I	I	I	I	O	O	I	I	I	I	I	I	I	I	I	I	O

(iv) Apply the recomposition operation on the selected sets.

Note that, in the new partition, the composite vectors should be removed form the subset $\underset{\sim}{\overline{\overline{\mathbf{N}}}}_a$ and possibly included in other subsets including the complete blocks of $\underset{\sim}{\overline{\overline{\mathbf{N}}}}_a$ and complete blocks created during the recomposition. The reason why the value of I_m is taken into account is that larger I_m's increase the probability of finding "recomposable" sets in the next iteration of the recomposition process.

Consider, for example, the vectors of Table 2 representing the implicit partition for the fifth outcome of the BWR example solved in Section 6.1. As explained above, recomposition operations are applied on bicontribution vectors. The vectors are sorted and divided into different sets according to their complete blocks. The procedure starts from the subset with the fewer complete blocks. This includes the vectors $\overline{\mathbf{L}}_{24}$ and $\overline{\mathbf{L}}_{26}$, which are merged to give $\overline{\mathbf{L}}_{24+26}$. Table 3 shows the Cartesian set updated with $\overline{\mathbf{L}}_{24+26}$, which can now be merged with $\overline{\mathbf{L}}_{28}$ and so forth. The recomposition choices are not always so few and they can be conflicting. The subset of Table 4 appears after a few iterations. A crude analysis indicates the potential of creating new complete blocks at:

(i) block b_4 by merging $\overline{\mathbf{L}}_{\text{i}}$ & $\overline{\mathbf{L}}_{\text{iii}}$ and/or $\overline{\mathbf{L}}_{\text{iv}}$ & $\overline{\mathbf{L}}_{\text{viii}}$,

(ii) block b_5 by merging $\overline{\mathbf{L}}_{\text{ii}}$ & $\overline{\mathbf{L}}_{\text{iii}}$, $\overline{\mathbf{L}}_{\text{v}}$ & $\overline{\mathbf{L}}_{\text{vii}}$ and/or $\overline{\mathbf{L}}_{\text{vi}}$ & $\overline{\mathbf{L}}_{\text{viii}}$,

(iii) block b_{10} by merging $\overline{\mathbf{L}}_{\text{v}}$ & $\overline{\mathbf{L}}_{\text{vi}}$ and/or $\overline{\mathbf{L}}_{\text{vii}}$ & $\overline{\mathbf{L}}_{\text{viii}}$.

The above seven merging actions involve binary state blocks, so their initial scores are equal to 2. This score is multiplied by the number of candidate merges per block (I_m), giving a score of 4 for the 4 merges in blocks b_4 and b_{10} and 6 for the 3 merges in block b_5. The possible merging actions are sorted according to their score and an action can take place if it does not conflict with any of the higher score actions. In this sense the proposed actions are $\overline{\mathbf{L}}_{\text{ii}}$ & $\overline{\mathbf{L}}_{\text{iii}}$, $\overline{\mathbf{L}}_{\text{v}}$ & $\overline{\mathbf{L}}_{\text{vii}}$ and $\overline{\mathbf{L}}_{\text{vi}}$ & $\overline{\mathbf{L}}_{\text{viii}}$.

This involvement of I_m in the score calculation stems from the observation that the resulting vectors have many common entries so the possibility of these vectors being treated as candidates for subsequent merges is very high. In effect, amongst the merged vectors of Table 5, vectors $\overline{\mathbf{L}}_{\text{v+vii}}$ and $\overline{\mathbf{L}}_{\text{vi+viii}}$ are candidates to be merged. However, this potential should be examined along with other vectors featuring the same complete blocks.

5.3. Algorithm Implementation. The decomposition and recomposition operations discussed above are each implemented into an iterative procedure. Following is a step-by-step description of the algorithm, using the motor-operated valve (MOV) example of Table 1 to illustrate the different procedures.

Step 1. Acquire event outcome data. This step returns a table of K columns for the K component blocks and 1 column for the outcomes. Following is the data array for the MOV example:

$$\begin{bmatrix} 1 & 1 & 1 & \\ 1 & & & \\ 1 & 1 & 2 & 4 \\ 1 & 1 & 3 & 1 \\ 1 & 1 & 4 & 1 \\ 1 & 2 & 1 & 3 \\ 1 & 2 & 2 & 3 \\ 1 & & & \\ 2 & 3 & 2 & \\ 1 & 2 & 4 & 3 \\ 1 & 3 & 1 & 4 \\ 1 & 3 & 2 & 4 \\ 1 & 3 & 3 & 1 \\ 1 & 3 & 4 & \\ 1 & & & \\ 1 & 4 & 1 & 3 \\ 1 & 4 & 2 & 3 \\ 1 & 4 & 3 & 2 \\ 1 & 4 & 4 & 2 \\ 2 & 1 & 1 & 1 \\ 2 & & & \\ 1 & 2 & 1 & \\ 2 & 1 & 3 & 1 \\ 2 & 1 & 4 & 1 \\ 2 & 2 & 1 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 3 & \\ 2 & & & \\ 2 & 2 & 4 & 2 \\ 2 & 3 & 1 & 1 \\ 2 & 3 & 2 & 1 \\ 2 & 3 & 3 & 1 \\ 2 & 3 & 4 & 1 \\ 2 & & & \\ 4 & 1 & 2 & \\ 2 & 4 & 2 & 2 \\ 2 & 4 & 3 & 2 \\ 2 & 4 & 4 & \\ 2 & & & \end{bmatrix}$$

Step 2. Get the contribution vectors referring to the outcome partition. The 32 events in columns 1 to 3 of the above array can be divided into 13, 11, 5, and 3 events according to the outcome they yield. The following arrays can be derived for the 4 outcomes:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 1 & 1 & 4 & 1 \\ 1 & 3 & 3 & 1 \\ 1 & 3 & 4 & 1 \\ 2 & 1 & 1 & 1 \\ 2 & 1 & 2 & 1 \\ 2 & 1 & 3 & 1 \\ 2 & 1 & 4 & 1 \\ 2 & 3 & 1 & 1 \\ 2 & 3 & 2 & 1 \\ 2 & 3 & 3 & 1 \\ 2 & 3 & 4 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 2 \\ 1 & 4 & 3 & 2 \\ 1 & 4 & 4 & 2 \\ 2 & 2 & 1 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 4 & 2 \\ 2 & 4 & 1 & 2 \\ 2 & 4 & 2 & 2 \\ 2 & 4 & 3 & 2 \\ 2 & 4 & 4 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 3 \\ 1 & 2 & 2 & 3 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 3 \\ 1 & 4 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 2 & 4 \\ 1 & 3 & 1 & 4 \\ 1 & 3 & 2 & 4 \end{bmatrix}$$

Using composition vectors, each of these arrays gives a row of the following array:

$$\overline{\overline{N^R}} = \begin{bmatrix} 5 & 8 & 7 & 0 & 6 & 0 & 3 & 2 & 4 & 4 & 1 \\ 3 & 8 & 0 & 5 & 0 & 6 & 2 & 2 & 4 & 3 & 2 \\ 5 & 0 & 0 & 3 & 0 & 2 & 2 & 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 & 2 & 0 & 1 & 2 & 0 & 0 & 4 \end{bmatrix}. \tag{11}$$

The $\overline{\overline{N^R}}$array now stores the contribution vector partition according to the system outcome. The superscript R is used to indicate that the last column of $\overline{\overline{N^R}}$ also stores the number of the outcome associated with each contribution vector.

Step 3. Check data consistency. This step exposes any irregularities present in the original data, by checking that all the possible complete events are present in the event table, only these events are present, and the table has no duplicate entries.

A fist check involves the columns of the contribution vector array. Based on the number of block states, we estimate the exact number of occurrences for each one of them and see if the sums of entries in the array satisfy this constraint. The MOV example has 3 blocks of 2, 4, and 4 states each. Therefore, each one of the 2 states of block 1 should occur 16 times; each one of the 4 states of blocks 2 and 3 should occur 8 times each. In effect, $16 = 5 + 3 + 5 + 3 = 8 + 8 + 0 + 0$; $8 = 7 + 0 + 0 + 1 = 0 + 5 + 3 + 0 = \cdots = 4 + 3 + 1 + 0$.

The second check involves the rows of the contribution vector array. Each entry refers to a specific state of a specific block. In each row, the total number of occurrences of the block 1 states should be equal to the number of occurrences of blocks 2 and 3. Looking at the first row of the contribution vector array $5 + 8 = 7 + 0 + 6 + 0 = 3 + 2 + 4 + 4$.

Step 3 is repeated for the other rows of $\overline{\overline{N^R}}$.

Step 4. Contribution vector decomposition. This step applies the decomposition operation (Section 4.1) and the decomposition rules (Section 5.1) to the first row of array $\overline{\overline{N^R}}$. Let, for instance, the contribution vector be $[5\ 8\ 7\ 0\ 6\ 0\ 3\ 2\ 4\ 4]$. The vector exhibits complete state blocks in block 1 (2nd state) and block 3 (3rd,

Table 3: Recomposition example: set of bicontribution vectors.

$\overline{\mathbf{L}}_i$	b_1				b_2			b_3		b_4		b_5		b_6		b_7		b_8		b_9		b_{10}	
24 + 26	O	O	O	I	I	O	O	I	O	O	I	O	I	**I**	**O**	I	O	I	O	I	I	I	O
28	O	O	O	I	I	O	O	I	O	O	I	O	I	**O**	**I**	I	O	I	O	I	I	I	O
21	I	O	O	O	I	O	I	I	I	O	I	O	I	I	O	I	O	I	O	I	O	I	O
25	I	O	O	O	I	O	I	I	I	O	I	O	I	I	O	I	O	I	O	O	I	I	O
11	O	O	O	I	I	O	O	I	O	I	O	I	O	I	I	I	O	I	O	I	I	I	O
…																							

Table 4: Recomposition example: subset of bicontribution vectors including only b_6, b_7, b_8, and b_9 as complete blocks.

$\overline{\mathbf{L}}_i$	b_1				b_2			b_3		b_4		b_5		b_6		b_7		b_8		b_9		b_{10}	
i	O	O	O	I	I	O	O	O	I	**O**	**I**	**I**	**O**	I	I	I	I	I	I	I	I	**I**	**O**
ii	O	O	O	I	I	O	O	O	I	**I**	**O**	**O**	**I**	I	I	I	I	I	I	I	I	**I**	**O**
iii	O	O	O	I	I	O	O	O	I	**I**	**O**	**I**	**O**	I	I	I	I	I	I	I	I	**I**	**O**
iv	O	O	O	I	I	O	I	O	O	**O**	**I**	**I**	**O**	I	I	I	I	I	I	I	I	**I**	**O**
v	O	O	O	I	I	O	I	O	O	**I**	**O**	**O**	**I**	I	I	I	I	I	I	I	I	**O**	**I**
vi	O	O	O	I	I	O	I	O	O	**I**	**O**	**O**	**I**	I	I	I	I	I	I	I	I	**I**	**O**
vii	O	O	O	I	I	O	I	O	O	**I**	**O**	**I**	**O**	I	I	I	I	I	I	I	I	**O**	**I**
viii	O	O	O	I	I	O	I	O	O	**I**	**O**	**I**	**O**	I	I	I	I	I	I	I	I	**I**	**O**

4th states). The rules indicate that the decomposition is first applied to block 1 and later to block 3. According to Section 5.1, the vector [5 8 7 0 6 0 3 2 4 4] is decomposed into the Cartesian contribution vector [0 8 4 0 4 0 2 2 2 2] and the remaining contribution vector [5 0 3 0 2 0 1 0 2 2].

Step 5. Update system arrays. There are 1 + R working arrays where contribution vectors (and their outcome) are stored. The first one is $\overline{\overline{N^R}}$. The others are denoted by $\overline{\overline{\underset{\sim}{N}}}_i$ and store the Cartesian vectors generated during Step 4 for each response m, starting from $i = 1$. Each row of $\overline{\overline{N^R}}$ treated during Step 4 is replaced by the remaining contribution vector. In our example, during the first iteration $\overline{\overline{N^R}}$ and $\overline{\overline{\underset{\sim}{N}}}_1$ become

$$\overline{\overline{N^R}} = \begin{bmatrix} 5 & 0 & 3 & 0 & 2 & 0 & 1 & 0 & 2 & 2 & 1 \\ 3 & 8 & 0 & 5 & 0 & 6 & 2 & 2 & 4 & 3 & 2 \\ 5 & 0 & 0 & 3 & 0 & 2 & 2 & 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 & 2 & 0 & 1 & 2 & 0 & 0 & 4 \end{bmatrix}, \qquad (12)$$

$$\overline{\overline{\underset{\sim}{N}}}_1 = [0 \;\; 8 \;\; 4 \;\; 0 \;\; 4 \;\; 0 \;\; 2 \;\; 2 \;\; 2 \;\; 2] .$$

Note that, there is no need to store outcome information in the arrays $\overline{\overline{\underset{\sim}{N}}}_i$.

Steps 4 and 5 are repeated until a decomposition operation leads to two Cartesian vectors. Then, both vectors are added to $\overline{\overline{\underset{\sim}{N}}}_i$, the first row of $\overline{\overline{N^R}}$ is removed, and i is replaced by $i + 1$.

Step 6. Decompose $\overline{\overline{\underset{\sim}{N}}}_i$. The array $\overline{\overline{\underset{\sim}{N}}}_i$ is further decomposed to get an array $\overline{\overline{\underset{\approx}{N}}}_i$ of implicit contribution vectors. Since the vectors have only Cartesian entries, the operations can be easily applied on the bicontribution vectors of $\overline{\overline{\underset{\sim}{N}}}_i$ and $\overline{\overline{\underset{\approx}{N}}}_i$, denoted by $\overline{\overline{\underset{\sim}{L}}}_i$ and $\overline{\overline{\underset{\approx}{L}}}_i$, respectively. The arrays referring to the contribution vector [5 8 7 0 6 0 3 2 4 4] are

$$\overline{\overline{\underset{\sim}{L}}}_1 = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad (13)$$

and, after further decomposition,

$$\overline{\overline{\underset{\approx}{L}}}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \qquad (14)$$

Step 6 supports the following recomposition actions.

Step 7. Apply recomposition actions. This step applies the recomposition operation (Section 4.2) and the recomposition rules (Section 5.2) to the parts of $\overline{\overline{\underset{\approx}{L}}}_i$ sharing the same outcome. Note that, the example considered here is too small and simple to offer potential for recomposition.

Steps 4–7 are repeated until the array $\overline{\overline{N^R}}$ is empty.

Step 8. Algorithm termination. The final output of the procedures described here is the arrays $\overline{\overline{\underset{\approx}{L}}}_m$, which represent implicit partitions of significantly reduced cardinality compared to the size of the system event table. Note that, the decomposition and the recomposition operations developed here ensure that the consistency of the data is preserved throughout the vector processing.

Table 5: Recomposition example: subset of bicontribution vectors including only the vectors appearing in Table 4.

$\overline{\mathbf{L}}_i$	b_1				b_2		b_3			b_4		b_5		b_6		b_7		b_8		b_9		b_{10}	
i	O	O	O	I	I	O	O	O	I	O	I	I	O	I	I	I	I	I	I	I	I	I	O
iv	O	O	O	I	I	O	I	O	O	O	I	I	O	I	I	I	I	I	I	I	I	I	O
v + vii	O	O	O	I	I	O	I	O	O	I	O	I	I	I	I	I	I	I	I	I	I	**O**	**I**
vi + viii	O	O	O	I	I	O	I	O	O	I	O	I	I	I	I	I	I	I	I	I	I	**I**	**O**
ii + iii	O	O	O	I	I	O	O	O	I	I	O	I	I	I	I	I	I	I	I	I	I	I	O

The Matlab environment is chosen as suitable for the fast manipulation of matrices using the built-in matrix operations. For instance, the decomposition when there are no Cartesian entries requires knowledge of the exact event subspace that corresponds to the processed contribution vector. The algorithm can either keep track of the events contributing in each vector or go through the original event table to isolate the subspace relating to each vector. The former, though it is more sophisticated, takes up a lot of memory even for relatively small problems. Matlab takes advantage of its built-in matrix operations for sort and find, to reduce significantly the execution time.

6. Case Studies

6.1. Case Study 1. The first case study is taken from Papazoglou [10] and concerns the development of an event tree for a boiling water nuclear reactor. The system involves 10 state blocks with 2 to 4 block states each. The event space consists of 3072 complete events and the system has 5 outcomes. Papazoglou [10] provided a set of Boolean equations and developed functional block diagrams that embedded information on the dependencies between the blocks. He finally presented a reduced event tree of 41 branches. Note that, if the reactor system is treated in BowTieBuilder [12, 13] without providing dependency information, the resulting event tree has 110 branches. This confirms that the efficiency of functional block diagram applications in reducing the size of event trees depends on the structure of the Boolean model, that dictates the dependencies between the blocks.

The methodology proposed here takes as input the original 3072×11 event table and produces the results reported in Table 6. Note that

(i) the states $\{1, 2, 3, 4\}$ of block I correspond to $\{\mathrm{L, M, N, T}\}$, and

(ii) the outcomes $\{1, 2, 3, 4, 5\}$ correspond to {CI, CII, CIII, CIV, Success}

of Papazoglou [10]. Each row of Table 6 gives a Cartesian vector (or an implicant) corresponding to a branch of the event tree. In this sense, the reduced tree described here has only 38 branches. The proposed algorithm identifies an inconsistency in the partition of block C of the original data. Resolving it leads to different results for outcome CIV, and this explains the three branches difference between 41 and 38. The rest of the branches/implicants are notably the same, with a single exception, involving the choice to expand block U rather than block Q (see bold cells of Table 6, lines 12–15). While both choices yield four branches/implicants, this differentiation shows that the procedure proposed here is not biased by the order of the blocks in the event table data.

6.2. Case Study 2. The proposed methodology is tested against a large problem involving 16 blocks, including one block having four state instances, four blocks with three states and eleven binary. The original event table has 663552×17 cells. The system has 5 possible outcomes. The initial event table is constructed via recursive partitions of the event subspace.

The resulting implicit partition has totally 273 vectors; in particular 86, 115, 34, 28, and 71 for the five outcomes. The recomposition stage requires 1.08 CPU seconds. Then, the final implicit partitions have totally 178 vectors; in particular 31, 54, 16, 20, and 57 for the respective partitions of the five outcomes.

CPU times can give an idea of the relative effort invested in the different activities taking place during a run. In this relatively large problem, the preparatory Steps 1–3 of Section 5.3 require 1.27 CPU seconds. The decomposition steps require only 0.0469 CPU seconds for a total of 86 decompositions using rule (a) and 124 CPU seconds for a total of 52 decompositions using rule (b) of Section 5.1. Therefore, the application of decompositions on the basis of Cartesian entries reduces the computational effort by almost 4 orders of magnitude. Clearly, an intelligent reduction of the frequency of visiting the event table would bring significant benefits in the computational times. Note that, CPU times refer to an Intel Core Quad 2.50 GHz processor with 1.95 GB RAM.

Finally, the proposed procedure manages to reduce the expanded event tree to just 0.0268% of its original size. The final partitions are easily translated into a set of implicants. There is no proof that this is a prime set, since there is lack of theoretical background on sufficient and necessary minimality conditions. In any case, the proposed methodology is a fast, effective, and intelligent way to reduce substantially a large event tree and facilitate the quantification of risk.

7. Conclusions

This work presents a new methodology for the reduction of event trees without the use of structural or functional information on the system. The work applies a holistic approach based on the concept of contribution vectors, to generate a minimal set of implicants representative of the system behavior. The method inherits the advantages and limitations of event tree representation. In this sense, the method is not hindered by component interdependences

Table 6: Final reduced event table for case study 1.

No. of event tree branch (or implicant)	Blocks states										Outcome r_i
	I	C	M	Q	U	X	E	I	V	W	
1	—	2	—	—	—	—	—	—	—	—	4
2	1	1	—	—	—	—	1	1	—	1	5
3	1	1	—	—	—	—	1	1	—	2	2
4	1	1	—	—	—	—	1	2	—	—	3
5	1	1	—	—	—	—	2	—	—	—	3
6	2	1	—	1	—	—	—	—	—	1	5
7	2	1	—	1	—	—	—	—	—	2	2
8	2	1	—	2	1	—	—	—	—	1	5
9	2	1	—	2	1	—	—	—	—	2	2
10	2	1	—	2	2	1	—	—	1	1	5
11	2	1	—	2	2	1	—	—	1	2	2
12	2	1	—	**2**	**2**	1	—	—	2	—	1
13	2	1	—	**2**	**2**	2	—	—	—	—	1
14	3	1	—	**—**	**1**	—	—	—	—	1	5
15	3	1	—	**—**	**1**	—	—	—	—	2	2
16	3	1	—	—	2	1	—	—	1	1	5
17	3	1	—	—	2	1	—	—	1	2	2
18	3	1	—	—	2	1	—	—	2	—	1
19	3	1	—	—	2	2	—	—	—	—	1
20	4	1	1	1	—	—	—	—	—	—	5
21	4	1	1	2	1	—	—	—	—	1	5
22	4	1	1	2	1	—	—	—	—	2	2
23	4	1	1	2	2	1	—	—	1	1	5
24	4	1	1	2	2	1	—	—	1	2	2
25	4	1	1	2	2	1	—	—	2	—	1
26	4	1	1	2	2	2	—	—	—	—	1
27	4	1	2	—	—	—	1	1	—	1	5
28	4	1	2	—	—	—	1	1	—	2	2
29	4	1	2	—	—	—	1	2	—	—	3
30	4	1	2	—	—	—	2	—	—	—	3
31	4	1	3	—	1	—	—	—	—	1	5
32	4	1	3	—	1	—	—	—	—	2	2
33	4	1	3	1	2	—	—	—	—	1	5
34	4	1	3	1	2	—	—	—	—	2	2
35	4	1	3	2	2	1	—	—	1	1	5
36	4	1	3	2	2	1	—	—	1	2	2
37	4	1	3	2	2	1	—	—	2	—	1
38	4	1	3	2	2	2	—	—	—	—	1

and noncoherent behavior in the considered systems. The proposed representation framework stems from Cartesian products to define partitions using composition vectors. The representation provides the basis for the application of decomposition and recomposition operations on single composition vectors and composition vector partitions. Implementation issues for the efficient use of these operations within an iterative algorithmic framework are discussed thoroughly.

The proposed method is tested against two case studies, one found in the literature and a fictitious large scale problem. In the former, the method provides a set of prime implicants very similar to the one reported in the literature. The latter illustrates the efficiency of the method in handling large-scale problems and proves the computational advantages from the proposed representation and operations.

Future work considers the use of the theoretical background presented here to develop necessary and sufficient conditions for the minimality of the final set of implicants. These conditions could then be incorporated in the recomposition stage to guide an optimal search algorithm towards the set of prime implicants.

Nomenclature

K:	Number of system blocks
b_k:	System block, $k \in \{1, 2, \dots, K\}$
$\mathbf{S}_k$:	Set of internal states of block b_k, $\mathbf{S}_k = \{s_k^1, s_k^2, \dots, s_k^{K_{S_k}}\}$
$s_k^{i_k}$:	i_kth internal state of block b_k, $i_k \in \{1, 2, \dots, K_{s_k^{i_k}}\}$
$\mathbf{E}$:	System event space
$\hat{e}$:	Complete joint event, $\hat{e} \in \mathbf{E}$
$\mathbf{R}$:	Set of the all possible system outcomes
r_j:	System outcome, $j \in \{1, 2, \dots, \vert\mathbf{R}\vert\}$
T:	Event table mapping $T : \hat{e} \mapsto r = T(\hat{e})$, where $\hat{e} \in \mathbf{E}$ and $r \in \mathbf{R}$
$\mathbf{A}, \mathbf{B}$:	Nonempty subspaces of $\mathbf{E}$
$\vert\mathbf{X}\vert$, $K_{\mathbf{X}}$:	Number of elements (cardinality) of set $\mathbf{X}$
$\mathbf{Q}(\mathbf{A})$:	Partition applied over $\mathbf{A}$
$\mathbf{q}_i^Q(\mathbf{A})$:	ith subset of $\mathbf{A}$ according to $\mathbf{Q}(\mathbf{A})$, $i \in \{1, 2, \dots, \mathbf{K}_{\mathbf{Q}(\mathbf{A})}\}$
$\mathbf{Q}^T(\mathbf{A})$:	Outcome-based partition of $\mathbf{A}$ (according to mapping $\mathbf{T}$)
$\mathbf{S}_k^{\mathbf{A}}$:	Set of b_k-block states, $k \in \{1, 2, \dots, K\}$, in the events comprising $\mathbf{A}$
$\mathbf{C}(\mathbf{A})$:	Cartesian product $\mathbf{S}_1^{\mathbf{A}} \times \mathbf{S}_2^{\mathbf{A}} \times \cdots \times \mathbf{S}_K^{\mathbf{A}}$
$\utilde{\mathbf{A}}$:	Cartesian subspace
$\underset{\approx}{\mathbf{A}}$:	Implicit subspace
$\utilde{\mathbf{Q}}(\mathbf{A})$:	Cartesian partition over subspace $\mathbf{A}$
$\utilde{\mathbf{q}}_i^{\mathbf{Q}}(\mathbf{A})$:	ith subset of $\utilde{\mathbf{Q}}(\mathbf{A})$, $i \in \{1, 2, \dots, K_{\utilde{\mathbf{Q}}(\mathbf{A})}\}$
$\underset{\approx}{\mathbf{Q}}(\mathbf{A})$:	Implicit partition over $\mathbf{A}$
$\underset{\approx}{\mathbf{q}}_i^{\mathbf{Q}}(\mathbf{A})$:	ith subset of $\underset{\approx}{\mathbf{Q}}(\mathbf{A})$, $i \in \{1, 2, \dots, K_{\underset{\approx}{\mathbf{Q}}(\mathbf{A})}\}$
$\overline{\mathbf{N}}(\mathbf{A})$:	Contribution vector of $\mathbf{A}$
$n_k^{i_k}(\mathbf{A})$:	Entry of $\overline{\mathbf{N}}(\mathbf{A})$, $n_k^{i_k}(\mathbf{A}) \in \mathbb{Z}_+$ and $k \in \{1, 2, \dots, K\}$, $i_k \in \{1, 2, \dots, K_{s_k^{i_k}}\}$
$\overline{\mathbf{L}}(\mathbf{A})$:	Bicontribution vector of subspace $\mathbf{A}$
$l_k^{i_k}(\mathbf{A})$:	Entry of $\overline{\mathbf{L}}(\mathbf{A})$, $l_k^{i_k}(\mathbf{A}) \in \{\mathrm{O}, \mathrm{I}\}$ and $k \in \{1, 2, \dots, K\}$, $i_k \in \{1, 2, \dots, K_{s_k^{i_k}}\}$
$\utilde{\overline{\mathbf{N}}}(\mathbf{A})$:	Cartesian contribution vector of $\mathbf{A}$
$\utilde{n}_k^{i_k}(\mathbf{A})$:	Entry of $\utilde{\overline{\mathbf{N}}}(\mathbf{A})$, $\underset{\approx}{n}_k^{i_k}(\mathbf{A}) \in \mathbb{Z}_+$ and $k \in \{1, 2, \dots, K\}$, $i_k \in \{1, 2, \dots, K_{s_k^{i_k}}\}$
$\overline{\overline{\mathbf{N}}}(\mathbf{Q}(\mathbf{A}))$:	Contribution vector partition of $\mathbf{Q}(\mathbf{A})$
$\overline{\mathbf{N}}(\mathbf{q}_i^{\mathbf{Q}}(\mathbf{A}))$:	ith member of $\overline{\overline{\mathbf{N}}}(\mathbf{Q}(\mathbf{A}))$, $i \in \{1, 2, \dots, K_{\mathbf{Q}(\mathbf{A})}\}$
λ:	Vector length
$\overline{Y}$:	Vector
$\overline{\overline{Y}}$:	Vector partition (i.e., set of vectors)
$\utilde{Y}$:	Entity obeying the Cartesian property
$\underset{\approx}{Y}$:	Entity obeying the property of implicitness
$\Lambda[\overline{\mathbf{N}}(\mathbf{A})]$:	Operation applied on $\overline{\mathbf{N}}(\mathbf{A})$ to obtain $\overline{\mathbf{L}}(\mathbf{A})$
$\Xi[\overline{\mathbf{L}}(\mathbf{A})]$:	Operation applied on $\overline{\mathbf{L}}(\mathbf{A})$ to obtain $\utilde{\overline{\mathbf{N}}}(\mathbf{A})$
σ_k:	Subset of $\mathbf{S}_k$
$\mathbf{A}^{\sigma_k}$:	Subset of $\mathbf{A}$ such that $\mathbf{A}^{\sigma_k} = \{\hat{e} \in \mathbf{A} \cap \Delta^{\sigma_k}\}$
$\Delta_{\mathbf{A}}^{\sigma_k}$:	Event subspace such that $\Delta_{\mathbf{A}}^{\sigma_k} = \{\mathbf{S}_1^{\mathbf{A}} \times \mathbf{S}_2^{\mathbf{A}} \times \cdots \times \sigma_k \times \cdots \times \mathbf{S}_K^{\mathbf{A}}\}$.

Glossary

Complete set of block states:	Set of all the possible states of a certain system block
Complete joint event:	Joint event containing an instance of each one of the system blocks
Cartesian property:	Event subspaces and contribution vectors that can be generated by a Cartesian product. Also, subspaces partitions and partition contribution vectors that can be generated by a set of Cartesian products
Property of implicitness:	Cartesian entities (i.e., subspaces, vectors, partitions) whose associated Cartesian products contain only complete or singleton sets of block states
Cartesian entries:	Nonzero entries of a contribution vector equal to their relative Cartesian contribution vector entries.

References

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

[2] J. Andrews, "System reliability modelling: the current capability and potential future developments," *Proceedings of the Institution of Mechanical Engineers C*, vol. 223, no. 12, pp. 2881–2897, 2009.

[3] I. A. Papazoglou and B. J. M. Ale, "A logical model for quantification of occupational risk," *Reliability Engineering and System Safety*, vol. 92, no. 6, pp. 785–803, 2007.

[4] R. Remenyte-Prescott and J. D. Andrews, "An efficient real-time method of analysis for non-coherent fault trees," *Quality and Reliability Engineering International*, vol. 25, no. 2, pp. 129–150, 2009.

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

[6] S. C. M. Rocco and M. Muselli, "Approximate multi-state reliability expressions using a new machine learning technique," *Reliability Engineering and System Safety*, vol. 89, no. 3, pp. 261–270, 2005.

[7] Y. Dutuit and A. Rauzy, "Approximate estimation of system reliability via fault trees," *Reliability Engineering and System Safety*, vol. 87, no. 2, pp. 163–172, 2005.

[8] W. S. Jung, J.-E. Yang, and J. Ha, "Development of measures to estimate truncation error in fault tree analysis," *Reliability Engineering and System Safety*, vol. 90, no. 1, pp. 30–36, 2005.

[9] I. A. Papazoglou, "Mathematical foundations of event trees," *Reliability Engineering and System Safety*, vol. 61, no. 3, pp. 169–183, 1998.

[10] I. A. Papazoglou, "Functional block diagrams and automated construction of event trees," *Reliability Engineering and System Safety*, vol. 61, no. 3, pp. 185–214, 1998.

[11] A. Rauzy and Y. Dutuit, "Exact and truncated computations of prime implicants of coherent and non-coherent fault trees within Aralia," *Reliability Engineering and System Safety*, vol. 58, no. 2, pp. 127–144, 1997.

[12] B. J. M. Ale, H. Baksteen, L. J. Bellamy et al., "Quantifying occupational risk: the development of an occupational risk model," *Safety Science*, no. 2, pp. 176–185, 2008.

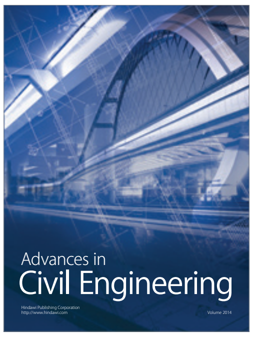
[13] A. R. Hale, B. J. M. Ale, L. H. J. Goossens et al., "Modeling accidents for prioritizing prevention," *Reliability Engineering & System Safety*, vol. 92, no. 12, pp. 1701–1715, 2007.

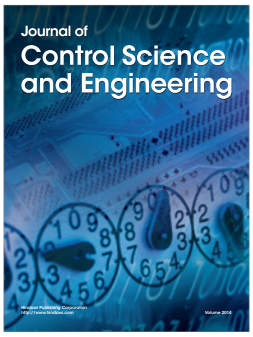
Journal of
Engineering

The Scientific World Journal

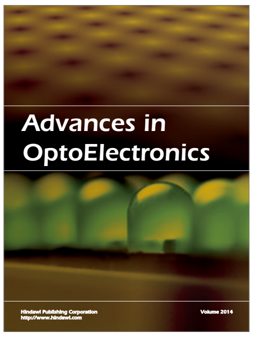
International Journal of
Rotating Machinery

Journal of
Sensors

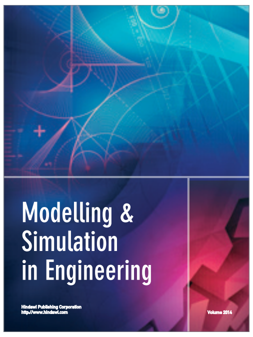
International Journal of
Distributed Sensor Networks

Advances in
Civil Engineering

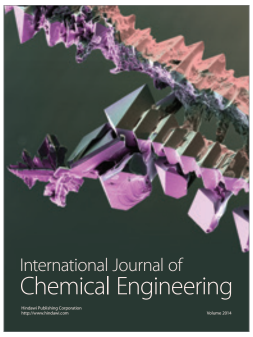
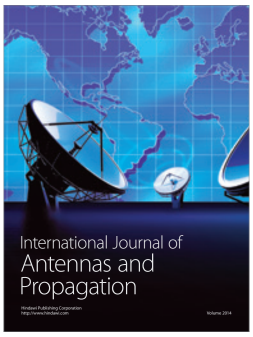

Journal of
Control Science and Engineering

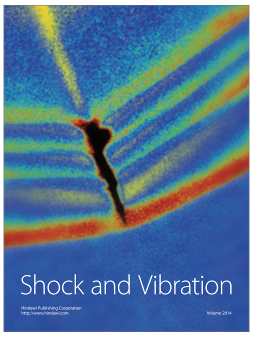
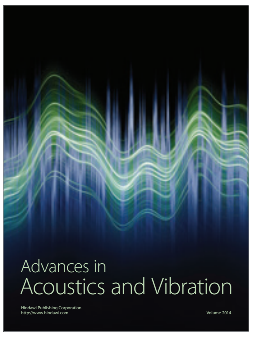
Journal of
Robotics

Journal of
Electrical and Computer Engineering

Advances in
OptoElectronics

VLSI Design

International Journal of
Navigation and Observation

Modelling & Simulation in Engineering

International Journal of
Aerospace Engineering

International Journal of
Chemical Engineering

International Journal of
Antennas and Propagation

Active and Passive Electronic Components

Shock and Vibration

Advances in
Acoustics and Vibration