*14th International Conference on Applications of Statistics and Probability in Civil Engineering, ICASP14*
*Dublin, Ireland, July 9-13, 2023*

# Efficient reliability analysis of coherent systems with discrete-state components

Ji-Eun Byun
*Lecturer, James Watt School of Engineering, Univ. of Glasgow, United Kingdom*
Daniel Straub
*Professor, Engineering Risk Analysis Group, Techniche Universität München, Germany*

ABSTRACT: Many engineering systems have their components modelled with discrete states. To calculate failure probabilities in such systems, a common approach is to employ decomposition algorithms that decompose an event space into two domains of system survival and failure. This study proposes a generalised decomposition algorithm that is applicable to systems with discrete component states and a coherent definition of system failure. The proposed algorithm raises several questions, which are investigated from both analytical and numerical perspectives.

## 1. INTRODUCTION

In many engineering systems, components are modelled to have a few discrete states to reflect reality more accurately. For example, in a transportation network, traffic capacity of roadways is modelled to have a few possible values to reflect failing of each lane. The reliability analysis of systems with discrete states can be challenging because they often contain a large number of components and because discontinuous and nonsmooth failure domains hinder the application of standard sampling-based methods for rare event estimation (Chan et al., 2022).

Given discrete component states, a common approach to reliability analysis is to decompose the event space of all components using minimal conditions of failure and survival (Lim and Song, 2012; Jane and Laih, 2008); in this paper, we term these conditions *failure rules* and *survival rules*[1]. This strategy exploits the fact that a system failure (survival) event is often determined by failures (survivals) of a small subset of components. For this strategy to be efficient, one should aim at (1) identifying minimal rules with the least number of system analyses and (2) identifying rules with higher probabilities first. The second goal is important if there are too many rules to identify all. In this case, one can obtain bounds on a failure probability.

Identifying minimal rules for an arbitrary system can require high computational cost. However, this cost can be alleviated when additional information on a system is taken into account. In this paper, we focus on *systems whose failure definition is coherent*, i.e. systems whose state does not deteriorate (improve) when component states are improved (worsened). We focus on coherency since it can significantly expedite identification of minimal rules, and many real-world systems satisfy this condition, at least approximately[2].

For reliability analysis of systems with discrete component states and a coherent definition of the failure event, this study proposes and systematically investigates a general decomposition algorithm to handle those systems. The paper is organised as follows: First, in Section 2 we introduce the key concepts and definitions and illustrate them with a simple example. Section 3 presents the pro-

[1]When component states are binary, rules are equivalent to the familiar concepts, *minimal link sets* and *minimal cut sets*.

[2]Example sources of incoherency are operational errors or bi-directional interactions between users.

posed algorithm and discusses its complexity. The proposed algorithm raises several questions, which are addressed in Section 4. Finally, we suggest topics for future research in Section 5 and conclude the discussions in Section 6.

## 2. CONCEPT

### *2.1. Definitions and notations*

In this section, we introduce the concept of minimal failure rules and minimal survival rules and provide definitions necessary for the remainder of the paper.

Let random variables $\boldsymbol{X} = \{X_1, \ldots, X_N\}$ represent the states of components. We refer to a combined state of all components, $\boldsymbol{x}$ as *a component vector state*. The system state is binary, i.e. either failure or survival. The *system function* $\Phi$ (aka structure function) outputs the system state in function of $\boldsymbol{x}$. Without loss of generality, it is assumed that a higher component state is associated with better functionality of the component.

Failure of the system is represented by a set of failure rules $\mathscr{F} = \{F_1, \ldots, F_Q\}$: A failure rule $F_q$ is defined by a tuple $\langle \boldsymbol{n}_q, \boldsymbol{r}_q \rangle$ where $\boldsymbol{n}_q \subset \{1, \ldots, N\}$ is a vector of indices of components that contribute to the $q$-th failure mode, and $\boldsymbol{r}_q$ denotes a state of these components that results in a failure of the system[3]. A failure rule $F_q = \langle \boldsymbol{n}_q, \boldsymbol{r}_q \rangle$ indicates that a system always fails given any component vector state $\boldsymbol{x} \leq \boldsymbol{r}_q$, regardless of the states of the components that are not in $\boldsymbol{n}_q$. A *minimal* failure rule is one in which any subtraction from $\boldsymbol{n}_q$ or any increase in $\boldsymbol{r}_q$ makes it no longer a failure rule. Similarly, a survival rule $S_q = \langle \boldsymbol{n}_q, \boldsymbol{r}_q \rangle \in \mathscr{S} = \{S_1, \ldots, S_Q\}$ indicates that a system survives given any component vector state $\boldsymbol{x} \geq \boldsymbol{r}_q$. A *minimal* survival rule is one in which any subtraction from $\boldsymbol{n}_q$ or any decrease in $\boldsymbol{r}_q$ makes it no longer a survival rule.

To simplify the presentation and discussion, the number of component states, which is denoted as $M$ in this paper, is considered constant across all component events although this constraint can be readily removed for implementation.

[3] Although $\boldsymbol{n}_q$ is a vector, when there is no confusion, it is also used as a set.

### *2.2. Illustrative example*

Consider the illustrative example of Figure 1. The network consists of five edges, which constitute the components, i.e. $N = 5$. Each edge has $M = 4$ states, $1, 2, 3, 4$ that correspond to flow capacities of $0, 1, 2, 3$ (i.e. component state 2 signifies a flow capacity of 1). All components are independent and have identical distributions such that $P(X_n = 1) = 0.01$, $P(X_n = 2) = 0.02$, $P(X_n = 3) = 0.03$ and $P(X_n = 4) = 0.94$. The system failure event is defined as the maximum flow from node 1 to 4 being less than 5 (when all edges have their maximum capacity, the maximum flow is 6).

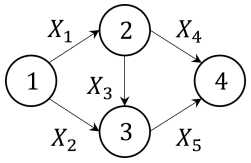



*Figure 1: Illustrative example*

This network has eight minimal failure rules and three minimal survival rules: $F_1 = \langle (1,5), (3,3) \rangle$, $F_2 = \langle (2), (2) \rangle$, $F_3 = \langle (4), (2) \rangle$, $F_4 = \langle (5), (2) \rangle$, $F_5 = \langle (1), (2) \rangle$, $F_6 = \langle (1,2), (3,3) \rangle$, $F_7 = \langle (4,5), (3,3) \rangle$, $F_8 = \langle (2,3,4), (3,1,3) \rangle$, $S_1 = \langle (1,2,4,5), (4,3,4,3) \rangle$, $S_2 = \langle (1,2,4,5), (3,4,3,4) \rangle$ and $S_3 = \langle (1,2,3,4,5), (4,3,2,3,4) \rangle$. For example, $F_1$ indicates that if $X_1$ and $X_5$ have states equal to or less than 3, the system always fails irrespective of the states of the other components. Similarly, $S_1$ indicates that if $X_1$, $X_2$, $X_4$ and $X_5$ have states equal to or greater than 4, 3, 4 and 3, respectively, the system always survives.

Given the rules and utilising the coherency property, one can decompose the outcome space of $\boldsymbol{X}$ to calculate the system failure probability. For example, $F_1$ implies that the component vector states satisfying $\boldsymbol{x} \leq (3,4,4,4,3)$ correspond to the system failure event (note that the maximum state is 4). In other words, $F_1$ represents a set of events whose probability amounts to $P(x_1 \leq 3) \cdot P(x_2 \leq 4) \cdot P(x_3 \leq 4) \cdot P(x_4 \leq 4) \cdot P(x_5 \leq 3) = 0.06 \cdot 1 \cdot 1 \cdot 1 \cdot 0.06 = 0.0036$. With all minimal rules considered, the system failure probability is calculated as 0.117. This is the probability of the union of all fail-

ure rules, whose exact computation becomes infeasible with a large number of components and rules. Hence we propose to evaluate the probability of this union via Monte Carlo Simulation in Section 3.2.

## 3. DECOMPOSITION ALGORITHM FOR RELIABILITY ANALYSIS WITH DISCRETE COMPONENT STATES

### *3.1. Proposed algorithm*

We propose a generalised decomposition algorithm consisting of the following steps:

- Step 1 Select a component vector state and evaluate the system function to determine the corresponding system state.
- Step 2 If the system state is survival (failure), identify a corresponding minimal survival (failure) rule. This can be done by subsequently lowering (raising) component states until any change alters the system state.
- Step 3 Evaluate bounds on the system failure probability using the identified minimal rules.
- Step 4 If the bounds on the system failure probability are sufficiently narrow, terminate the algorithm. Otherwise, return to Step 1 to narrow the bounds or go to Step 5 to perform additional analysis on the unidentified component vector states.
- Step 5 If necessary, perform further analysis on the set of component vector states that are not covered by the minimal failure and survival rules. Such an analysis can be performed, e.g. via sampling or surrogate modelling.

In each step of the algorithm, alternative approaches are possible. These should be selected to minimise the number of system analyses for evaluating the system failure probability to a desired accuracy. In the following, we discuss and investigate some of the possible approaches.

### *3.2. Approximation of event space using Monte Carlo simulation*

Given a set of failure rules, a lower bound to the system failure probability is given by the probability of the union event of all given rules, while a rule is an intersection event of component events. This results in an exponential increase in computational cost with respect to the number of rules and the number of components in rules[4]. Since both factors increase rapidly as the number of components increases, this calculation becomes infeasible even with a moderate number of components. To avoid this issue, previous decomposition algorithms are mostly developed to identify mutually exclusive rules (Lim and Song (2012); Jane and Laih (2008)). However, those algorithms do not reduce underlying computational complexity as they create as many rules as the number of calculations required for corresponding non-exclusive rules.

To address this challenge, we propose approximating the sample space of $\boldsymbol{X}$ by a set of Monte Carlo (MC) samples. This approach enables one to circumvent an expensive analytical calculation of probabilities. Provided a sufficiently large number of MC samples, this strategy leads to a sufficiently accurate solution with known accuracy.

Specifically, in the beginning of the proposed decomposition algorithm, one can generate a large number of MC samples. Then, a sample is classified as failure if there is any failure rule whose component states are all equal to or greater than the sample (i.e. that *dominates* the sample), as survival if there is any survival rule that dominates the sample, and, otherwise, as unknown. Then, if all samples have been assigned to the system failure or survival event, the system failure probability can be estimated by counting the number of failure samples; otherwise, approximate bounds on the probability can be evaluated by counting the number of samples in each of the three categories. The set of unassigned samples also serves as potential starting points in Step 1.

In this strategy, the order of calculations to process identified rules is $O(N_{\mathrm{MCS}} \cdot (Q_f + Q_s) \cdot N)$, where $N_{\mathrm{MCS}}$, $Q_f$, $Q_s$ and $N$ are respectively the number of MC samples, that of failure rules, that of survival rules and that of component events. In other words, the order of computational complex-

[4]Precisely, the number of calculations is $\prod_{q=1}^{Q} |\boldsymbol{n}_q|$ given a set of failure rules $\mathscr{F} = \{F_1, \ldots, F_Q\}$, where $|\boldsymbol{n}_q|$ denotes the length of $\boldsymbol{n}_q$.

ity is decreased from exponential to pseudo-linear. Note that these calculations do not involve any evaluations of the system function, hence $N_{\mathrm{MCS}}$ can be chosen quite high.

### *3.3. Complexity of rule identification*

Given $Q_f$ minimal failure rules and $Q_s$ minimal survival rules, the total number of system analyses required to identify all rules is

$$N_a = (Q_f + Q_s) \cdot \mathbb{E}[N_{a1}], \tag{1}$$

where $\mathbb{E}[N_{a1}]$ is the average number of system analyses required to identify one rule. In the following subsections, we investigate how the terms in the equation depend on the size and definition of a system.

#### *3.3.1. Number of minimal rules*

The numbers of minimal rules, $Q_f$ and $Q_s$, depend on two factors: (1) the number of components and (2) the definition of a system event. The number of rules usually increases rapidly with the number of components, whose worst case is on exponential order. However, such increase can be much milder when a system event has a simple definition. For example, for series or parallel systems, the number of minimal rules increases linearly with the number of components (Byun et al., 2019).

To investigate how the number of minimal rules depends on system definition, we consider a system performance $Y = \sum_{n=1}^{N} \alpha_n X_n$, where the binary component events $X_1, \ldots, X_N$ take either state 1 or 0. The system failure event is defined as $Y \leq \beta \cdot \sum_{n=1}^{N} \alpha_n$. Then, three settings are investigated: (1) all components have identical weights $\alpha_n = 1$, $\forall n$, (2) 50% of components have higher weights as $\alpha_n = 5$, $n = 1, \ldots, N/2$ and $\alpha_n = 1, n = N/2+1, \ldots, N$, and (3) 10% of components have higher weights as $\alpha_n = 10, n = 1, \ldots, N/10$ and $\alpha_n = 1$, $n = N/10+1, \ldots, N$. Meanwhile, to investigate the effect of $\beta$, we examine three values 0.1, 0.25 and 0.5.

The results are summarised in Figure 2, where, for comparison, the line $2^N$ (i.e. the total number of component vector states) is also drawn. One can see that the larger the $\beta$, the greater the number of rules is. This shows that the number of minimal rules decreases when either the failure rules or the survival rules prevail over the others. Second, particularly when $\beta$ is away from 0.5, components having identical importance lead to a smaller number of rules. Nevertheless, this does not mean that identical components enable one to identify all minimal rules as their number still rapidly increases with the number of components (note that the y-axis is in logarithmic scale).

#### *3.3.2. Number of analyses for identifying a rule*

In Step 2 of the algorithm, we see two possible approaches to identifying the minimal rules: (1) always start from the best (worst) state if a given sample is a survival (or failure) event and (2) perform a binary search. The first approach would be preferred when minimal rules mostly involve a very small subset of components as it will need only one system analysis for all other components. In contrast, the second approach is preferred when most minimal rules involve most of the component events as it is known to be most efficient in general search problems. It is noted that difference between the two approaches is minor given a small number of component states (e.g. less than 5).

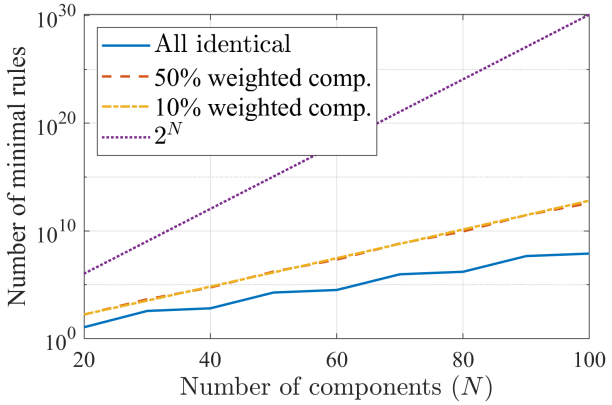
We compare the two approaches by the required number of system analyses to identify a rule $\langle \mathcal{N}_q, \boldsymbol{r}_q \rangle$ from a vector of component states $\boldsymbol{x}$. The first approach starting from the worst/best states requires

$$N_{a1,w} = (N - N_q) + \sum_{n'=1}^{N_q} r_{n'}, \tag{2a}$$

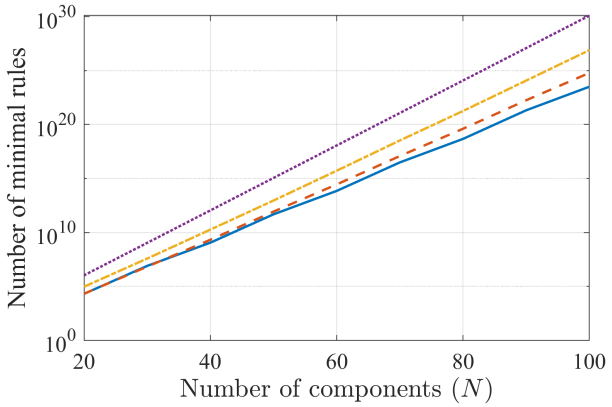
system evaluations in the case of a survival rule and

$$N_{a1,w} = (N - N_q) + \sum_{n'=1}^{N_q} (M - r_{n'} + 1), \tag{2b}$$

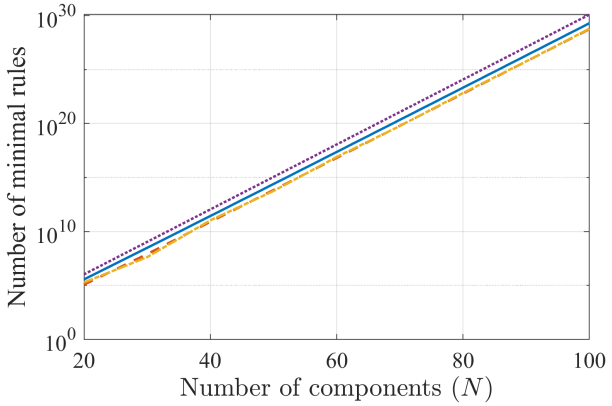
in the case of a failure rule. The first term is derived from the fact that only one analysis is required for components $n \notin \boldsymbol{n}_q$. The second term is the required number of analyses for $n' \in \boldsymbol{n}_q$, which depends on how far $r_{n'}$ is from the worst/best state. It is noted that Eq. (2) does not depend on start points as long as the points lead to the same rule $\boldsymbol{r}_q$.




*(a)* $\beta = 0.1$




*(b)* $\beta = 0.25$




*(c)* $\beta = 0.5$

*Figure 2: The number of minimal rules with varying settings*

The second search approach requires

$$N_{a1,b} = \sum_{\substack{n\in\{1,\dots,N\},\\ n\neq \boldsymbol{n}_q}} (\lfloor \log_2 x_n \rfloor + 1) + \sum_{n'=\boldsymbol{n}_q} (\lfloor \log_2 (x_{n'} - r_{n'} + 1) \rfloor + 1), \tag{3a}$$

system evaluations in the case of a failure rule and

$$N_{a1,b} = \sum_{\substack{n\in\{1,\dots,N\},\\ n\neq \boldsymbol{n}_q}} (\lfloor \log_2 (M - x_n + 1) \rfloor + 1) + \sum_{n'=\boldsymbol{n}_q} (\lfloor \log_2 (r_{n'} - x_{n'} + 1) \rfloor + 1), \tag{3b}$$

in the case of a survival rule. In this case, the numbers depend on how far apart $\boldsymbol{x}$ and $\boldsymbol{r}_q$ are.

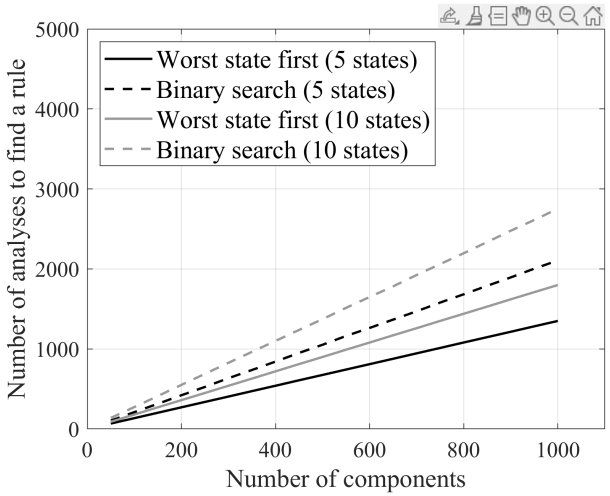
For numerical investigation of Eq. (2) and Eq. (3), we calculate the required number of system analyses by generating 100 samples of starting points and minimal rules. Starting points are generated by assuming uniform distributions over component states. We consider two cases with $M = 5$ and $M = 10$ states per component. Since failure rules are likely to consist of lower states of components, failure rules are generated from a uniform distribution from state 1 to state 2 for $M = 5$ and from state 1 to state 3 for $M = 10$.

The mean number of system evaluations to find a minimal rule are shown in Figure 3. Considering that $N_q$ is likely to be much less than $N$, we investigate two cases where $N_q = 0.1N$ and $N_q = 0.5N$. As expected, it is found that when $\alpha_q$ is low (Figure 3a), the worst/best-first search is more efficient, while the binary search requires less system analyses when $\alpha_q$ is relatively high (Figure 3b). The results are consistent over both cases of $M = 5$ and $M = 10$. The results also suggest the potential of a hybrid approach that one first investigates a worst/best state, and if further analysis is required, a binary search is opted over the rest of the states.

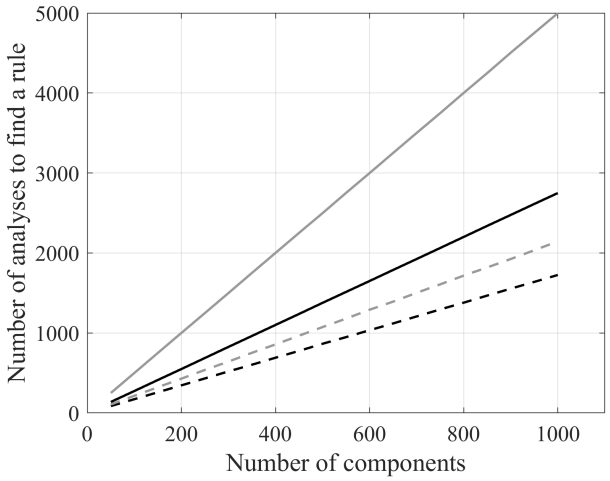
## 4. ADDITIONAL OBSERVATIONS AND INVESTIGATIONS

### *4.1. Investigation order of components matters when interest lies in bounds of a probability.*

A component vector state is typically included in multiple minimal rules. Therefore, different orderings of components during decomposition can lead to the identification different minimal rules. When one intends to identify all minimal rules, the identification order of rules does not affect the number of system analyses. However, if the purpose is to obtain bounds on the system failure probability, identifying rules with higher probabilities reduces the




*(a)* $N_q = 0.1N$.




*(b)* $N_q = 0.5N$.

*Figure 3: Sample means of the required number of system analyses to identify a rule.*

number of system analyses. A general observation is that rules with higher probabilities are often those consisting of a smaller number of components and component states with higher probabilities. This observation can be used to order components; for example, one can first examine components with higher probability and/or components that appear more often in previously found rules. Such strategies are used in (Lim and Song, 2012) and (Lee and Song, 2021).

### 4.2. *Narrowing down bounds is easier when components have disparate importance.*

In Section 3.3.1, we found that having components with identical importance results in a smaller number of minimal rules. However, when the interest lies in narrowing down the bounds on a system failure probability (rather than calculating its exact value), the bounds might be narrowed down more efficiently when components have disparate importance rather than equal importance. To show this, we return to a case of the experiment in Section 3.3.1 with $N = 100$ and $\beta = 0.25$. Figure 4 shows a bar chart of the rule lengths, where the $y$-axis shows the proportions of each rule length (i.e. the number of components in a rule).

The figure shows that a higher variation in components' importance leads to a higher variance in minimal rule lengths. For systems with large component vector states, only a very small proportion of rules can be identified, which might be even less than the leftmost bars in the figure. In such cases, the third case (10% weighted components) can narrow down bounds at the highest rate as its shortest minimal rules would have higher probabilities than those of the other cases. For example, if component events $X_1, \ldots, X_n$ are statistically independent, and $P(X_n = 0) = 0.1$, one more component in a minimal rule leads to a 10% reduction of the rule's probability.

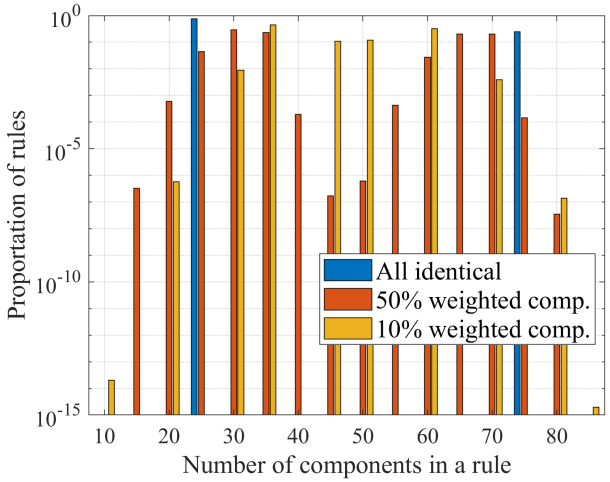



*Figure 4: Proportion of rule lengths with* $N = 100$ *and* $\beta = 0.25$

### 4.3. *The starting point is not critical.*

This section investigates the influence of the choice of starting component vector state on the number of system function evaluations required to

identify a corresponding minimal rule. To this end, we set up an experiment to test how the distance between a starting vector and a corresponding minimal rule affects the number of system function evaluations. Following the notations in Section 3.3.2, parameters are set to $N_q = 0.3N$ and $M = 5$, and the maximum state in a rule is set to 3, while the number of components is varied from 50 to 1,000. Then, two settings of the distance between a starting point and a rule are compared: each component state can differ (1) by only 1 state and (2) by any number of states. Since a closed-form analysis is unavailable, we generate 100 samples over the difference of component states by assuming a uniform distribution over possible numbers of state difference.

The sample means of the analysis numbers are illustrated in Figure 5. As observed in (2), the first approach does not depend on a starting point, making the two solid curves coincide with each other. The second approach does depend on a starting point, but the plot shows that this difference is insignificant. This shows that selection of the starting component vector state does not play a critical role.

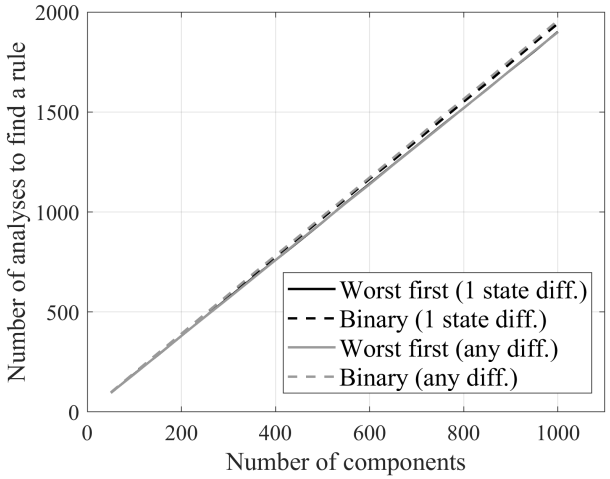



*Figure 5: Number of required analyses to identify a rule–with close and distant start points.*

### *4.4. The algorithm can identify all minimal rules.*

We argue that given a sufficiently large set of MC samples, the proposed algorithm can identify all existing minimal rules. This statement is equivalent to a set of two statements: (1) having no unknown samples (i.e. samples with unknown system state) indicates that all existing rules have been found, and (2) any unknown sample leads to a new minimal rule. While the first statement is true by construction, the second statement can be proved as follows: Consider a hitherto unknown sample $\boldsymbol{x}$, which turns out to be a survival event. Since it is an unknown sample, for each known rule $S_q \in \mathscr{S}$, there is at least one $n \in \boldsymbol{n}_q$ such that $x_n < r_{q(n)}$. Note that the algorithm never finds a survival rule that has a greater component state than the sample that it started from. Therefore, the identified rule will have at least one $n \in \boldsymbol{n}_q$ such that $x_n < r_{q(n)}$ for each of the known survival rules in $\mathscr{S}$. Hence, the new rule is not dominated by any known survival rule and, therefore, is a new rule. The same proof applies to when $\boldsymbol{x}$ is a failure event.

### *4.5. Components not in a survival rule can often be easily identified.*

In reality, it is often the case that given a sample of system survival, one can easily identify components that do not belong to a minimal survival rule from the result of a system analysis. This is because most system functions return the functionality status of each component, and the components that are not functioning are not part of the corresponding survival rule. For instance, in connectivity or maximum flow analysis, an analysis returns flows that take place on each component, and components that are not carrying any flow can be ruled out for search from the beginning (Lim and Song (2012); Jane and Laih (2008)). In such cases, the number of analyses required to identify a rule is reduced by the amount of the first term in Eq. (2b) and Eq. (3b).

Finding irrelevant components in a failure sample is less straightforward; we are not aware of system functions that would facilitate this. If possible, it would have the effect of reducing the required number by the amount of the first term in Eq. 2a and Eq. (3a).

## 5. FUTURE RESEARCH DIRECTIONS

Substantial improvements of the proposed algorithm are expected in two parts of the algorithm: (1) ordering of component events in identifying rules with higher probabilities (Step 2) and (2) inference

over the unassigned event space (Step 5). Improvements are possible either by utilising previously identified rules or by exploiting problem-specific information.

One can make the assumption that unidentified minimal rules have similar properties to the identified ones. For example, one can use the known rules to identify components with higher importance (i.e. those that appear in many rules) and start investigation from those components. Such approaches could be generally applicable. Their disadvantage is that general-purpose algorithms rarely outperform algorithms that utilise problem-specific information. Hence it can be worthwhile to explore problem-specific algorithms.

Existing examples of problem-specific algorithms include those developed for network connectivity (Lim and Song (2012)) and network flows (Jane and Laih (2008)). For instance, Lim and Song (2012) improves an existing decomposition algorithm for connectivity reliability by modifying Dijkstra's shortest path algorithm to identify survival rules having higher probability first. This is possible by intervening in the system function. However, how to perform such intervention is not evident for more complex analyses such as traffic simulation or power flow analysis.

Another possible research direction is the development of algorithms to predict the system state of unassigned component vector states, which is a classification problem. Achieving high prediction accuracy remains challenging with discrete component events and a large number of components (i.e. high-dimensional data). Opportunities for improving this lie in the development of (1) a distance metric (between assignments) that effectively handles the high dimensionality of data and (2) a probabilistic prediction that correctly quantifies uncertainties in prediction.

## 6. CONCLUSIONS

This paper considers reliability analysis of systems whose component events have discrete states and which has a coherent definition of failure and survival. To this end, we propose a generalised decomposition algorithm and quantify its computational complexity (i.e. the number of required system analyses to calculate a system failure probability). We present in-depth discussions on how to minimise the complexity of the proposed algorithm, for which several arguments are drawn from analytical derivations, numerical experiments and observations from earlier works. Finally, we suggest topics for future research.

## ACKNOWLEDGEMENTS

The first author acknowledges support by the Alexander von Humboldt Fellowship.

## REFERENCES

Byun, J.-E., Zwirglmaier, K., Straub, D., and Song, J. (2019). "Matrix-based bayesian network for efficient memory storage and flexible inference." *Reliability Engineering & System Safety*, 185, 533–545.

Chan, J., Papaioannou, I., and Straub, D. (2022). "An adaptive subset simulation algorithm for system reliability analysis with discontinuous limit states." *Reliability Engineering & System Safety*, 225, 108607.

Jane, C.-C. and Laih, Y.-W. (2008). "A practical algorithm for computing multi-state two-terminal reliability." *IEEE Transactions on reliability*, 57(2), 295–302.

Lee, D. and Song, J. (2021). "Multi-scale seismic reliability assessment of networks by centrality-based selective recursive decomposition algorithm." *Earthquake Engineering Structural Dynamics*, 50(8), 2174–2194.

Lim, H. and Song, J. (2012). "Efficient risk assessment of lifeline networks under spatially correlated ground motions using selective recursive decomposition algorithm." *Earthquake Engineering & Structural Dynamics*, 41(13), 1861–1882.