Proc. of the 8th Intl. Symp. on Reliability Engineering and Risk Management (ISRERM 2022)
Edited by Michael Beer, Enrico Zio, Kok-Kwang Phoon, and Bilal M. Ayyub

doi: 10.3850/978-981-18-5184-1_MS-14-136-cd

A new method on system reliability analysis based on survival signature theory

Yide Zheng[1], Yi Zhang*,[1], Jiarui Lin[1]

[1] *Department of Civil Engineering, Tsinghua University, China*

Abstract

Simulation method is widely used in system reliability analysis and not affected by the dimension of state function. But traditional simulation method will need large number of samples to obtain more accurate results when the failure probability of each component is quite low. Survival signature theory divides the system reliability into probability structure and system structure, which makes the calculation process easier and clearer. However, the assumption that components are independently and identically distributed (idd) is very idealistic which hinders the further application of this method. This paper presents two methods to calculate system reliability based on survival signature theory system with independent but nonidentical components and dependent components. This paper mainly discuss the complex calculation of nonidentical components and the exchangeability assumption of related components.

Keywords: system reliability, survival signature, divide and conquer, weighted random sampling, copula, dependent

1. Introduction

System reliability analysis is of great significance in engineering projects, especially for large complex systems including power network, water supply network and communication network. These systems typically have many components that are strongly correlated. Many system analysis approaches have been developed, for examples, fault tree (FT) model, binary decision diagram (BDD), Bayesian networks (BNs). These methods could identify the failure modes of the system and calculate the corresponding reliability. However, the modelling and reliability analysis of complex systems, which cannot be simply divided into a combination of series and parallel subsystems or with diverse external load conditions, are very challenging. Coolen and Coolen-Maturi had developed the survival signature approach, which can analyse complex systems with several types of components and inherits all the merits from system signature [1]. Survival signature theory can also work with arbitrary dependent structures between failure events. But the exchangeability assumption still must be satisfied in survival signature theory, and in most application scenarios, it is often replaced by the stronger assumption of component independent and identical (iid) distribution [2],[3],[4]. The following is a brief introduction to survival signature theory and its fundamentals. The following is a brief introduction to survival signature theory and its fundamentals.

Suppose a system consists of m components with K types of subsystems. Each subsystem has m_k components and $\sum_{k=1}^{K} m_k = m$. The state vector $\underline{x} = (x_1, x_2, \dots, x_m) \in \{0,1\}^m$ is defined to describe the working state for all components while $x_i = 1$ if the i-th component functions and $x_i = 0$ if not. $\phi = \phi(\underline{x}): \{0,1\}^m \to \{0,1\}$ defines the system structure function while $\phi(\underline{x}) = 1$ if the system fuctions and $\phi(\underline{x}) = 0$ if not. For such a system, the survival signature is denoted by $\Phi(l_1, l_2, \dots, l_K)$ with $l_k = 0,1,2, \dots, m_k$ for $k = 1,2, \dots, K$, which is defined to be the probability that the system functions given that l_k of its m_k components of type k work, for each $k \in \{1,2, \dots, K\}$. There are $\binom{m_k}{l_k} = \frac{m_k!}{(m_k - l_k)! \cdot l_k!}$ state vectors $\underline{x^k} = (x_1^k, x_2^k, \dots, x_{m_k}^k)$ with l_k components working in subsystem k. Let $S_{l_1,l_2,\dots,l_K}$ denotes the set of all state vectors for the whole system when l_k components working in the subsystem k ($k = 1,2, \dots, K$). Assume all the components are fully independent and components in the same subsystem are identically distributed (iid, independent and identically distributed), the survival signature can be calculated by the formula below

$$\Phi(l_1, l_2, \dots, l_K) = \left[\prod_{k=1}^{K} \binom{m_k}{l_k}^{-1}\right] \times \sum_{\underline{x} \in S_{l_1,l_2,\dots,l_K}} \phi(\underline{x}). \quad (1)$$

Let $C_k(t) \in \{0,1,2, \dots, m_k\}$ denote the number of k type components working at time t and the components in the same type subsystem have a known cumulative distribution function (CDF), $F_k(t)$, then:

$$P\left(\bigcap_{k=1}^{K}\{C_k(t) = l_k\}\right) = \prod_{k=1}^{K} P(C_k(t) = l_k) = \prod_{k=1}^{K} \binom{m_k}{l_k} [F_k(t)]^{m_k - l_k} [1 - F_k(t)]^{l_k}. \quad (2)$$

Hence, the survival function of the system with K types of components can be formulated as:

$$P(T_s > t) = \sum_{l_1=0}^{m_1} \dots \sum_{l_K=0}^{m_K} \Phi(l_1, l_2, \dots, l_K) P\left(\bigcap_{k=1}^{K}\{C_k(t) = l_k\}\right). \quad (3)$$

In Eq. (3), the survival signature $\Phi(l_1, l_2, \dots, l_K)$ is also called system structure while $P(\bigcap_{k=1}^{K}\{C_k(t) = l_k\})$ is the probability structure in the survival function. The system structure is completely determined by the composition of the system and the system failure mode in the iid assumption. The probability structure describes the probability of the occurrence of different state vectors. In this paper, we discuss about two situation that survival signature theory might be used in reliability analysis, system with independent but nonidentical components and system with dependent components.

2. Algorithm for systems with independent but nonidentical components

2.1 Algorithm for computing probability structure

When the components are nonidentical, the probability structure can no longer be expressed in the Bernoulli form of Eq. (2). Divide and conquer [5] refers to a class of algorithm techniques in which the target problem is broken into several parts, solves the problem in each part

* E-Mail: zhang-yi@tsinghua.edu.cn

recursively, and then combines the solutions to these subproblems into an overall solution. As shown in Eq. (2), it is quite cumbersome to calculate $C_{m_k}^{l_k}$ different equations (each of these equations is multiplied by m_k component probabilities) one-by-one when m_k or l_k gets larger. Nevertheless, the computation can be much easier if it is divided into several minor groups of calculations. For example, the recursive equation based on Bernoulli binomial in Eq. (4) shows the relationship between the system with m_k components and two divided subsystems with $m_k/2$ components

$$C_{m_k}^{l_k} = \sum_{i=0}^{l_k} C_{\frac{m_k}{2}}^{i} \cdot C_{\frac{m_k}{2}}^{l_k - i}. \quad (4)$$

In consideration of the characteristics of Bernoulli polynomials and the computational efficiency, each decomposition is dichotomized. However, not every partition will divide m_k into two sub-problems of $\frac{m_k}{2}$ size. Flow chart in Fig. 1 is used to solve the size of minimum divide-and-conquer units. Here, C represents the size of minimum divide-and-conquer units, the initial value is m_k; C_0 represents the upper limit of the size of minimum divide-and-conquer units; when the size of minimum divide-and-conquer units is less than C_0, no further division is performed; n represents the number of divisions.

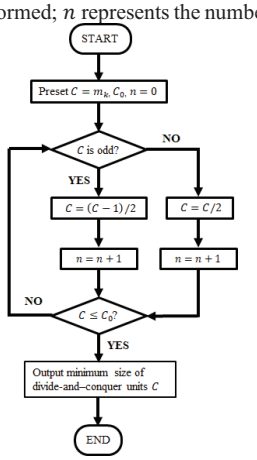

Figure 1. Flow chart to calculate the size of minimum divide-and-conquer units

The above algorithm can divide m_k components into units of size C and $C+1$. Suppose N_1 represents the number of units of size C and N_2 represents the number of units of size $C+1$, they satisfy the following relationship:

$$N_1 + N_2 = 2^n. \quad (5)$$

$$N_1 \cdot C + N_2 \cdot (C+1) = m_k. \quad (6)$$

By solving these equations, the number of division steps in the divide-and-conquer algorithm can be obtained. Here, an $(n+1) \times 2^n$ (n represents the number of division) matrix A is utilized to store every solution in the divide-and-conquer algorithm. The probability structure can firstly be solved with the minimum unit of division. The $(n+1)$th row of matrix A represents the probability structure of 2^n minimum divide-and-conquer units, each of which is a $1 \times (l_k+1)$ vector, and the jth element of the vector represents the probability that j components function in each minimum unit. If l_k is larger than the unit size C or $C+1$, the probability that j $(j = C, C+1, \dots, l_k)$ components in the unit work is 0. The matrix A and A $(n+1,1)$ are shown in Fig. 2a and Fig. 2b.

	1	2	3	4	...	2^{i-1}	...	2^n
1	$\{1 \times (l_k+1)\}$							
2	$\{1 \times (l_k+1)\}$	$\{1 \times (l_k+1)\}$						
3	$\{1 \times (l_k+1)\}$	$\{1 \times (l_k+1)\}$	...	$\{1 \times (l_k+1)\}$				
⋮	⋮	⋮	⋮	⋮	⋱			
i	$\{1 \times (l_k+1)\}$	...	...	...	...	$\{1 \times (l_k+1)\}$		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	
$n+1$	$\{1 \times (l_k+1)\}$	...	...	...	...	...	...	$\{1 \times (l_k+1)\}$

Figure 2a. Diagram of matrix A

	1	2	...	l_k+1
1	$P(C_{n_1}(t)=0)$	$P(C_{n_1}(t)=1)$	...	$P(C_{n_1}(t)=l_k)$

Figure 2b. Diagram of matrix $A(n+1,1)$

Note* in $P\left(C_{i_j}(t)=m\right)(i=1,\dots,n),(j=1,2,\dots,2^i)$, $(m=0,1,\dots,l_k)$represents the probability that m components work in the jth divided unit after i divisions.

The ith row of matrix A has 2^{i-1} vectors, each of which is derived from $(i+1)$th row by solving Eq. (7). For example, the first vector in nth row of the matrix A is derived from the first and second vectors in $(n+1)$th row, and the second vector in nth row is derived from the third and fourth vectors in $(n+1)$th row, ..., the 2^{n-1}th vector in nth row is derived from the (2^n-1)th and 2^nth vectors in $(n+1)$th row. The $A(i,j)$ is given as follows.

$$P\left(C_{i_j}(t)=s\right) = \sum_{q=0}^{s} P\left(C_{i+1_{2\times j-1}}(t)=q\right) \cdot P\left(C_{i+1_{2\times j}}(t)=s-q\right). \quad (7)$$

where $P\left(C_{i_j}(t)=s\right)(i=1,\dots,n),(j=1,2,\dots,2^i),(s=0,1,\dots,l_k)$ means the probability that s components work in the jth divided unit after i divisions, $P\left(C_{i+1_{2\times j-1}}(t)=q\right)$ means the probability that q components work in the $(2j-1)$ th divided unit after $i+1$ divisions, $P\left(C_{i+1_{2\times j}}(t)=s-q\right)$ means the probability that $s-q$ components work in the $(2j)$th divided unit after $i+1$ divisions. Flow chart of proposed algorithm in computing the probability structure is shown below.

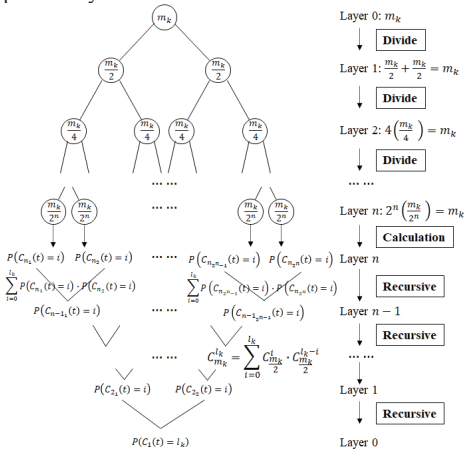

Figure 3. Flow chart of proposed algorithm

2.2 Algorithm for computing system structure

A proposed new simulation method for survival signature also adopts the idea of divide-and-conquer. Sampling is difficult for large m_k, thus, it is more practical to do the sampling in small groups, i.e., divided unit. As mentioned in Section 3.1, computing the probability of $l_k$component working in subsystem k can be converted into the combination of computing two probabilistic structures containing $\frac{m_k}{2}$ components assuming m_k is even, as shown in Eq. (7). For a sample with l_k component working in subsystem k, if we divide the m_k once, there will be $l_k + 1$ cases, that is, q $(q = 0,1, \dots, l_k)$ components working in the first divided part containing $\frac{m_k}{2}$ components while $l_k - q$ components working in the second divided part containing $\frac{m_k}{2}$ components. If we continue to divide the first divided part containing $\frac{m_k}{2}$ components with q $(q = 0,1, \dots, l_k)$ components working into 2 subparts, there will be another $q+1$ cases where $r(r = 0,1, \dots, q)$ components working in the first divided subpart containing $\frac{m_k}{4}$ components and $q - r$ components working in the second divided subpart containing $m_k/4$ components. Therefore, in each Monte Carlo sampling, we only need to determine how many components working in each of the two subparts formed by each division, as shown in Fig. 4. A matrix B (Fig. 5) is used to store the number of working components in each partition unit of a sample, where $l_{k_i-j}(i = 1, \dots, n), (j = 1,2, \dots, 2^i)$ represents the number of components working in the j th subsystem after i divisions for one generated sample, $l_{k_i-j} + l_{k_i-j+1} = l_{k_{i-1}-(j+1)/2}(j = 1,3,5, \dots 2^i - 1)$.

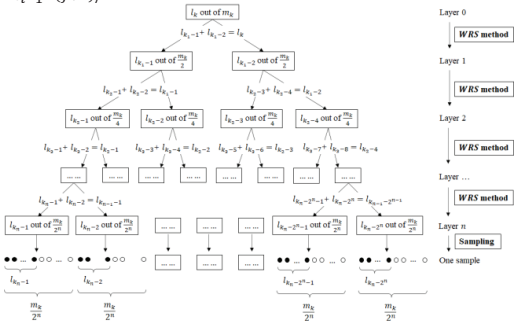

Figure 4. Flow chart of one generated l_k-out-of-m_k sample

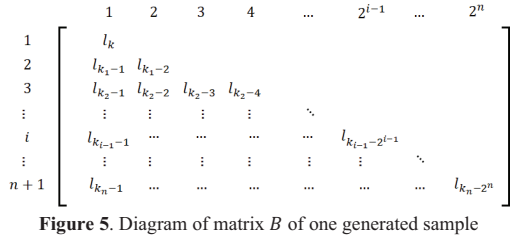

Figure 5. Diagram of matrix B of one generated sample

Matrix B is generated from top to bottom, $B(i+1, 2j)$ $(i = 1,2, \dots, n; j = 1,2, \dots 2^{i-1})$ and $B(i+1, 2j+1)$ are randomly sampled from $B(i, j) + 1$ cases in which $B(i, j)$ components working in the jth unit after $i - 1$ divisions. However, the weight of each case is different, which is related to the number of working components $B(i, j)$. The calculation formula is given as

$$\omega_{q+1} = \frac{P\left(C_{i_{2\times j-1}}(t) = q\right) \cdot P\left(C_{i_{2\times j}}(t) = B(i,j) - q\right)}{\sum_{z=0}^{B(i,j)} P\left(C_{i_{2\times j-1}}(t) = z\right) \cdot P\left(C_{i_{2\times j}}(t) = B(i,j) - z\right)}. \quad (8)$$

where ω_{q+1} represents the $(q+1)$th case weight $(q = 0,1, \dots, B(i, j))$, $B(i, j)$ represents the number of working components in the j th unit after $i-1$ divisions, $P\left(C_{i_{2\times j-1}}(t) = q\right)$ means the probability that q components work in the $(2j-1)$th divided unit after i divisions, $P\left(C_{i_{2\times j}}(t) = s - q\right)$ means the probability that $B(i, j) - q$ components work in the $(2j)$th divided unit after i divisions. So, the generation of matrix B is a weighted random sampling (WRS) process which is to randomly select m elements in a set V with n elements, $v_i \in V$ with a weight ω_i, and the probability of each element being selected is the ratio of the weight. Efraimidis [6],[7] proposed weighted random sampling algorithm to solve this problem, herein, the process of solving for $B(i+1, 2j)(i = 1,2, \dots, n; j = 1,2, \dots 2^{i-1})$ and $B(i+1, 2j+1)$ transforms into select a case from a set V with $B(i, j) + 1$ cases, each case having a weight $\omega_{q+1}(q = 0,1, \dots B(i, j))$.

Each element in the probability structure matrix A (Fig.2a) calculated by the divide and conquer algorithm represents probability that i $(i = 0,1, \dots, l_k)$ components work in the divided part. Each item in Eq. (8) can be obtained from matrix A to avoid repeated calculation. Different from the bottom-up calculation process of the divide-and-conquer algorithm, this sampling method is top-down. By selecting the generation mode of each layer, the last generated samples of the bottom layer are combined into the final samples. Therefore, there will be no invalid samples.

To calculate $\varphi(\underline{X})$, it has to determine whether generated state vector corresponds to system working or failure. The binary decision diagram (BDD) is introduced here to obtain the system state. The BDD is a directed acyclic graph (DAG) based on the Shannon decomposition. BDD can be obtained by decision tree transformation. Each BDD structure contains two terminal nodes labelled either with 0, or with 1, and one root node representing the first node in the chosen order. Different decision tree component selection order will lead to different BDD structure complexity [8]. It is difficult to traverse all the failure modes of the graph structure or find the optimal order of BDD structure modelling components, but Hardy [9] proposed a method based on the decomposition of the graph structure to build a system BDD structure. After obtaining the system BDD structure, it is very convenient to obtain the working state of the system by inputting the working state of each component. The pseudocode of the simulation method combining *WRS* algorithm and *BDD* structure to compute $\Phi(l_1, \dots, l_K)$ is shown below, where N_{work} represents the number of samples in which the system works out of N samples generated. The pseudocode of the simulation method combining *WRS* algorithm and *BDD* structure to compute $\Phi(l_1, \dots, l_K)$ is shown below, where N_{work} represents the number of samples in which the system works out of N samples generated.

Algorithm: Compute $\Phi(l_1, l_2, \dots, l_K)$

Input: generated N samples with l_k components working in subsystem k $(k = 1,2, \dots, K)$ *BDD* structure for system

Output: $\Phi(l_1, \dots, l_K)$

1: Set $N_{work} = 0$;

2: For n = 1 to N do

3: Put the generated $\underline{X_n} = (X_1, \dots, X_m)$ into *BDD* structure and get the terminal node $T_{\underline{X}}$;

4: If $T_{\underline{X}} = 1$, $N_{work} = N_{work}$+1, End-If

5: End-For

6: $\Phi(l_1, \dots, l_K) = \frac{N_{work}}{N}$

2.3 Illustrative example

Firstly, a simple system shown in Fig. 6 is selected as an example study herein. It can be regarded as a subsystem k separated from a complex system with m_k components. Components in the subsystem k are independent but nonidentical and the probability structure $P(C_k(t) = l_k)$ has to be determined. The effect of the size of the minimum divide and conquer unit on the computation time of the algorithm is analysed herein. Suppose $m_k = 960$, $l_k = 3$, the minimum divide and conquer unit size having 480, 240, 120, 60, 30, 15, 7 and 8 is tested separately. Figure 7 shows the relationship between calculation time and the number of divisions in divide and conquer algorithm. It can be seen the minimum divide-and-conquer unit greatly influent the survival signature computation time. The calculation time of the divide-and-conquer algorithm after 7 times of division (about 0.044273 sec) is the least. The calculation time for 7 divisions (about 0.044273 sec) is about 2000 times that for 1 division (about 108.954 sec). When the number of divisions $n = 7$, it takes about 0.000818 sec to calculate the minimum size of divide-and-conquer units, about 0.040147 sec to solve the solution for the minimum units, which shows that the main running time of the divide-and-conquer algorithm in calculation of probability structure is the conquering of the minimum divide-and-conquer unit, rather than the process of division and regression. And when we perform 8 divisions and the size of the minimum divide-and-conquer unit is 3 and 4, the running time of the algorithm is 0.026962 sec, and the overall improvement efficiency is not high. Therefore, the upper limit of the minimum partition and conquer unit $C_0 = 10$ could obtain better operation efficiency.

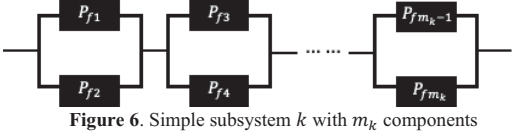

Figure 6. Simple subsystem k with m_k components

In order to prove the effectiveness of the algorithm, the divide and conquer algorithm is compared with the exhaustive algorithm. As mentioned in Section 2.1, there are $C_{m_k}^{l_k} = \frac{m_k!}{(m_k - l_k)! \cdot l_k!}$ cases when l_k components working in a m_k components system with different probabilities. The exhaustive algorithm refers to iterating over all the cases and calculating the probability of each case, then adding them up. By increasing m_k or l_k, the exhaustive algorithm leads to dramatic computations, for it will take long time to iterate all $C_{m_k}^{l_k}$ cases. The comparison of calculation time between exhaustive algorithm and proposed algorithm for different working component l_k for $m_k = 30$ with number of divisions equals to 2 is shown in Fig. 8. For $l_k = 3$, the comparison of calculation time between exhaustive algorithm and proposed algorithm for different m_k is shown in Fig. 9.

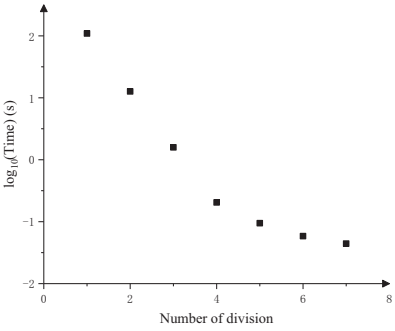

Figure 7. Relationship between calculation time and the number of divisions in divide-and-conquer algorithm

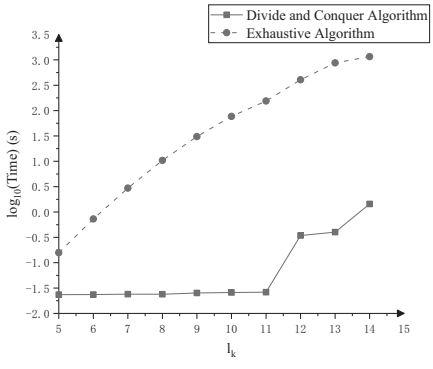

Figure 8. Comparison between exhaustive algorithm and divide and conquer algorithm in the calculation time for $m_k = 30$ when the number of divisions $n = 2$

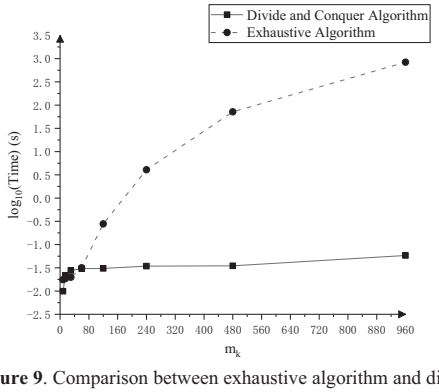

Figure 9. Comparison between exhaustive algorithm and divide and conquer in calculation time for $l_k = 3$

Then, the same type of series-parallel system in Fig. 6 with $m_k = 50$ is used to compare the new simulation method and the crude Monte Carlo simulation method (sample all the state vector sets according to the failure

probability of each component, and then select the state vectors satisfying the work of l_k components in subsystem k). Four groups of state vectors are tested, these are 40-out-of-50 ($m_k = 50$, $l_k = 40$), 25-out-of-50, 90-out-of-100, 50-out-of-100, 190-out-of-200 and 100-out-of-200 samples. Tab. 1 shows the sampling completion time of the two sampling methods. As can be seen from Tab.1, the new simulation method greatly improves the calculation time compared with the crude Monte Carlo method. And with the increase of the total number of components m_k and the number of failure components, the crude Monte Carlo method will not be able to collect samples effectively. This is mainly because the crude Monte Carlo needs to judge whether each generated sample belongs to set $S_{l_1,l_2,\ldots,l_K}$, which will inevitably produce invalid samples that do not meet the conditions, leading to the reduction of sampling efficiency. If the reliability of components decreases or the m_k and l_k of the system increases, the number of effective samples decreases as the number of effective samples from Monte Carlo simulations is about $N_{all} \times P(C_k(t) = l_k)$. When N_{all} remains unchanged and $P(C_k(t) = l_k)$ decreases, the number of effective samples decreases. It takes about 12 seconds for the crude Monte Carlo method to generate 10,000 valid samples for a total of 180,000 samples when $m_k = 50$, $l_k = 40$. For the new simulation method, the main work of the algorithm is to construct the B matrix of each sample, that is, to solve the weight of each case in each step of the weighted random sampling. Therefore, the running time of the algorithm is roughly $O(2^n \cdot l_k)$, where n denotes the number of divisions and l_k represents the number of working components in subsystem k.

Table 1. Comparison between new simulation method and crude Monte Carlo.

Case	Crude Monte Carlo	Proposed method
$m_k = 50$, $l_k = 25$	-	3.057s
$m_k = 50$, $l_k = 40$	11.456s	2.196s
$m_k = 100$, $l_k = 50$	-	7.278s
$m_k = 100$, $l_k = 90$	30.240s	3.590s
$m_k = 200$,$l_k = 100$	-	14.185s
$m_k = 200$, $l_k = 190$	-	7.705s

Next, the new developed approach is tested for analysing components having time-varying reliability properties. It is assumed that the failure probability of each component follows the Gamma distribution over time as given in Eq. (9) [10]. Intervals are used to describe the imprecision in the failure time distribution, with distribution parameter α in [1.2,1.8] and distribution parameter β in [2.3,2.9].

$$p_f(t) = \frac{\beta^\alpha}{\Gamma(\alpha)} t^{\alpha-1} e^{-\beta t} \tag{9}$$

where α and β are the shape parameter and size parameter for Gamma distribution, $\Gamma(\alpha)$ is the Gamma function while $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx$. The state vector generated by the new simulation method is converted into the BDD structure, and the survival signature values for different states are calculated and shown in Fig. 10. For a serial-parallel system with m_k components, it is easy to estimate the survival signature of l_k components working under idd assumption as shown in Eq. (10).

$$\Phi(l_k) = \frac{m_k \cdot (m_k - 2) \cdot (m_k - 3) \cdots \left(m_k - 2(l_k - 1)\right)}{m_k \cdot (m_k - 1) \cdot \cdots \cdot (m_k - l_k + 1)}. \tag{10}$$

Instead of the BDD structure, the system functions of each state vector can be obtained more simply by determining whether each parallel component fails simultaneously. Figure 10 shows the variation of survival signature (Φ(45), Φ(40), Φ(35)) over time when each component is independently but not identically distributed and the corresponding value of survival signature under idd assumption (solid lines shown in Fig. 10, in idd condition, Φ(45)=0.8024, Φ(40)=0.3258, Φ(35)=0.04758). It can be found that survival signature is no longer a fixed value as components are not identically distributed, which leads to different failure modes.

It should be realized the probability structure obtained by the divide and conquer algorithm is the analytical solution, and the system structure obtained by ***Algorithm*** is the simulation solution. In order to exclude the possibility that the difference in survival signature under the two conditions in Fig. 10 is due to the randomness of Monte Carlo sampling, three reliability calculation methods of the series-parallel system are compared.

1. Analytical method $r_1(t)$;
2. Survival signature method $r_2(t)$: based on Eq. (3), the system structure is calculated under the assumption of idd, and the probability structure is calculated by the divide and conquer algorithm;
3. Survival signature method $r_3(t)$: based on Eq. (3), the system structure is calculated by ***Algorithm***, and the probability structure is calculated by the divide and conquer algorithm.

Tab.2 shows the system reliability in each year calculated by each method.

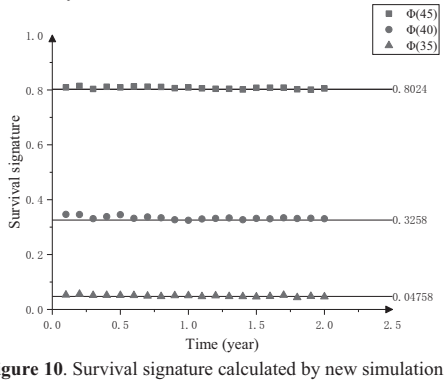

Figure 10. Survival signature calculated by new simulation method

Table 2. System failure probability calculated by each method and calculation error.

Time (year)	$r_1(t)$	$r_2(t)$	$r_3(t)$
0	1	1(0%)	1(0%)
0.1	0.95605	0.95251(0.3703%)	0.95616(0.0115%)
0.2	0.83188	0.8244(0.8992%)	0.83254(0.0793%)
0.3	0.65969	0.65044(1.4022%)	0.66221(0.382%)
0.4	0.47917	0.47037(1.8365%)	0.47921(0.00835%)
0.5	0.32073	0.31369(2.195%)	0.3222(0.4583%)
0.6	0.19904	0.19411(2.4769%)	0.19983(0.3969%)
0.7	0.11521	0.11211(2.6907%)	0.11525(0.0347%)
0.8	0.06254	0.06076(2.8462%)	0.06357(1.6469%)
0.9	0.03201	0.03107(2.9366%)	0.03171(0.9372%)
1.0	0.01552	0.01505(3.0284%)	0.01541(0.7088%)
1.1	0.00716	0.00694(2.8571%)	0.00707(1.257%)

1.2	0.00315	0.00306(3.0075%)	0.00313(0.6349%)

As can be seen from Tab. 2, the error obtained by method 2 is larger method 3. The results indicate that survival signature is no longer a fixed value in the case of independent but nonidentical components. Thus, the system structure calculated by method 2 is wrong. It can also be found that the reliability obtained by the new simulation method (method 3) is close to the analytical solution. This provides a solution for the reliability analysis of large complex systems when analytical solution is difficult to obtain.

5. Analytical reliability for systems with dependent components

5.1 Copula theory

The multidimensional copula function $C(u_1, u_2, \dots, u_n)$ refers to the function that satisfies the following properties:

(1) The domain of $C(u_1, u_2, \dots, u_n)$ is $[0,1]^n$;

(2) $C(u_1, u_2, \dots, u_n)$ is grounded and n-dimensional increasing;

(3) For every $u_i \in [0,1](i = 1,2, \dots, n)$, $C(1, \dots, 1, u_i, 1, \dots, 1) = u_i$.

Suppose $x_1, x_2, \dots, x_n$ are random variables, $F_1(x_1), F_2(x_2), \dots, F_n(x_n)$ are continuous one-dimensional distribution functions. Let $u_1 = F_1(x_1)$, $u_2 = F_2(x_2)$, …, $u_n = F_n(x_n)$, and $F(x_1, x_2, \dots, x_n)$ is a multidimensional joint cumulative distribution function with cumulative marginal distribution function $F_1(x_1), F_2(x_2), \dots, F_n(x_n)$. Multidimensional distribution Sklar theorem [11] states that there exists a copula function $C(u_1, u_2, \dots, u_n)$ that joins $F(x_1, x_2, \dots, x_n)$ with $F_1(x_1), F_2(x_2), \dots, F_n(x_n)$, as shown in Eq. (11):

$$F(x_1, x_2, \dots, x_n) = C[F_1(x_1), F_2(x_2), \dots, F_n(x_n)] = C(u_1, u_2, \dots, u_n) \quad (11)$$

The multidimensional Archimedean copula function is expressed as follows:

$$C(u_1, u_2, \dots, u_n; \varphi_\theta) = \varphi_\theta^{-1}(\varphi_\theta(u_1) + \varphi_\theta(u_2) + \cdots + \varphi_\theta(u_n)) \quad (12)$$

where θ is correlation parameter; $\varphi_\theta(\cdot)$ is the generator of Archimedean copula function; $\varphi_\theta^{-1}(\cdot)$ is the pseudo-inverse function of generator function.

5.2 Proof of exchangeability under dependent component situation

Suppose there are m_k components in subsystem k, $x_i(i = 1,2, \dots, m_k)$ represents the limit state function for component $i(i = 1,2, \dots, m_k)$. $x_i < 0$ refers to component i failure, while $x_i \geq 0$ refers to component i working. Here, for systems with dependent components, it is defined that components of type k have the same marginal cumulative distribution function $F_k(x)$, where $F_k(0)$ represents the failure probability for type k components. In the proving process, $F_k(x)$ can also be replaced by the time-varying failure distribution $F'_k(t)$, which means the probability that component fails less than time t. Let $f(x_1, x_2, \dots, x_{m_k})$ be the multidimensional joint probability density distribution, and $F(X_1, X_2, \dots, X_{m_k})$ be the multidimensional joint cumulative distribution function, which represents the probability that $x_1 \leq X_1, \dots, x_{m_k} \leq X_{m_k}$. If the correlation of components in the subsystem can be described by Archimedean copula function, then,

$$F(X_1, X_2, \dots, X_{m_k}) = \int_{-\infty}^{X_1}\int_{-\infty}^{X_2}\cdots\int_{-\infty}^{X_{m_k}} f(x_1, x_2, \dots, x_{m_k})dx_{m_k}\dots dx_2 dx_1 . \quad (13)$$

$$P(x_1 \leq X_1, \dots, x_{m_k} \leq X_{m_k}) = F(X_1, X_2, \dots, X_{m_k}) = C(u_1, u_2, \dots, u_n; \varphi_\theta) = \varphi_\theta^{-1}(\varphi_\theta(u_1) + \varphi_\theta(u_2) + \cdots + \varphi_\theta(u_n)). \quad (14)$$

where X_i and $u_i(i = 1,2, \dots, m_k)$ satisfy in Eq. (15) and Eq. (16),

$$u_i = F_i(X_i) = F(X_i). \quad (15)$$

$$X_i = F_i^{-1}(u_i) = F^{-1}(u_i). \quad (16)$$

where $F(x_i) \in [0,1]$ and $F^{-1}(\cdot)$ is the inverse function of the marginal cumulative distribution function, the corresponding $F^{-1}(\cdot)$ of each component is the same in the same subsystem. Now consider two components i and j $(i, j = 1,2, \dots, m_k; i \neq j)$ in the subsystem k, then the probability while components i working, components j failure and the probability while components i failure, components j working are shown in Eq. (17) and Eq. (18):

$$\begin{aligned}
&P(x_1 \leq X_1, \dots, x_i \leq 0, \dots, x_j > 0, \dots, x_{m_k} \leq X_{m_k}) \\
&= \int_{-\infty}^{X_1}\cdots\int_{-\infty}^{0}\cdots\int_{0}^{\infty}\cdots\int_{-\infty}^{X_{m_k}} f(x_1, x_2, \dots, x_{m_k})dx_{m_k}\dots dx_1 \\
&= \int_{-\infty}^{X_1}\cdots\int_{-\infty}^{0}\cdots\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{X_{m_k}} f(x_1, x_2, \dots, x_{m_k})dx_{m_k}\dots dx_1 \\
&- \int_{-\infty}^{X_1}\cdots\int_{-\infty}^{0}\cdots\int_{-\infty}^{0}\cdots\int_{-\infty}^{X_{m_k}} f(x_1, x_2, \dots, x_{m_k})dx_{m_k}\dots dx_1 \\
&= C(F_1(X_1), \dots, F_i(0), \dots, 1, \dots, F_{m_k}(X_{m_k}); \varphi_\theta) \\
&- C(F_1(X_1), \dots, F_i(0), \dots, F_j(0), \dots, F_{m_k}(X_{m_k}); \varphi_\theta) \\
&= \varphi_\theta^{-1}\begin{pmatrix}\varphi_\theta(F_1(X_1)) + \cdots + \varphi_\theta(F_i(0)) + \cdots \\ \varphi_\theta(1) + \cdots + \varphi_\theta\left(F_{m_k}(X_{m_k})\right)\end{pmatrix} \\
&- \varphi_\theta^{-1}\begin{pmatrix}\varphi_\theta(F_1(X_1)) + \cdots + \varphi_\theta(F_i(0)) + \cdots \\ \varphi_\theta\left(F_j(0)\right) + \cdots + \varphi_\theta\left(F_{m_k}(X_{m_k})\right)\end{pmatrix}.
\end{aligned} \quad (17)$$

$$\begin{aligned}
&P(x_1 \leq X_1, \dots, x_i > 0, \dots, x_j \leq 0, \dots, x_{m_k} \leq X_{m_k}) \\
&= \int_{-\infty}^{X_1}\cdots\int_{0}^{\infty}\cdots\int_{-\infty}^{0}\cdots\int_{-\infty}^{X_{m_k}} f(x_1, x_2, \dots, x_{m_k})dx_{m_k}\dots dx_1 \\
&= \int_{-\infty}^{X_1}\cdots\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{0}\cdots\int_{-\infty}^{X_{m_k}} f(x_1, x_2, \dots, x_{m_k})dx_{m_k}\dots dx_1 \\
&- \int_{-\infty}^{X_1}\cdots\int_{-\infty}^{0}\cdots\int_{-\infty}^{0}\cdots\int_{-\infty}^{X_{m_k}} f(x_1, x_2, \dots, x_{m_k})dx_{m_k}\dots dx_1 \\
&= C(F_1(X_1), \dots, 1, \dots, F_j(0), \dots, F_{m_k}(X_{m_k}); \varphi_\theta) \\
&- C(F_1(X_1), \dots, F_i(0), \dots, F_j(0), \dots, F_{m_k}(X_{m_k}); \varphi_\theta) \\
&= \varphi_\theta^{-1}\begin{pmatrix}\varphi_\theta(F_1(X_1)) + \cdots + \varphi_\theta(1) + \cdots \\ \varphi_\theta\left(F_j(0)\right) + \cdots + \varphi_\theta\left(F_{m_k}(X_{m_k})\right)\end{pmatrix} \\
&- \varphi_\theta^{-1}\begin{pmatrix}\varphi_\theta(F_1(X_1)) + \cdots + \varphi_\theta(F_i(0)) + \cdots \\ \varphi_\theta\left(F_j(0)\right) + \cdots + \varphi_\theta\left(F_{m_k}(X_{m_k})\right)\end{pmatrix}.
\end{aligned} \quad (18)$$

The components in the same type have the same marginal cumulative distribution probability, so $F_i(0) = F_j(0)$, and the generator function of the Archimedean copula is the same for each component, then it can be obtained that:

$$P(x_1 \leq X_1, \dots, x_i \leq 0, \dots, x_j > 0, \dots, x_{m_k} \leq X_{m_k}) = P(x_1 \leq X_1, \dots, x_i > 0, \dots, x_j \leq 0, \dots, x_{m_k} \leq X_{m_k}). \quad (19)$$

Similarly, if $F_k(x)$ is replaced by the time-varying failure distribution $F'_k(t)$, the exchangeability assumption could also be satisfied in Eq. (20):

$$P(t_1 \le T_1, \dots, t_i \le T, \dots, x_j > T, \dots, t_{m_k} \le T_{m_k}) = P(t_1 \le T_1, \dots, t_i > T, \dots, t_j \le T, \dots, t_{m_k} \le T_{m_k}). \quad (20)$$

In Eq. (19), when there are l_k components working in subsystem k, the probability of the $\binom{m_k}{l_k}$ possible cases are the same with each other. Hence, the probability structure that $l_k (k = 1,2, \dots, K)$ components working in subsystem k can be calculated in the following formula:

$$P(C_k = l_k) = \binom{m_k}{l_k} \cdot P(x_1 \ge 0, \dots, x_{l_k} \ge 0, x_{l_k+1} < 0, \dots, x_{m_k} < 0). \quad (21)$$

The above formula is no different from Eq. (4), except that the analytic expression of Bernoulli distribution is replaced by the Archimedean copula form. Therefore, we only need to compute the probability of one case in $S_{l_1,\dots,l_K}$ to obtain the probability structure.

5.3 Calculation of probability structure

In this part, a method for calculating $P(x_1 \ge 0, \dots, x_{l_k} \ge 0, x_{l_k+1} < 0, \dots, x_{m_k} < 0)$ in Eq. (21) is introduced. When $x_i < 0 (i = 1,2, \dots, m_k)$, according to the Sklar theorem, it can be converted to $F_i(0)$ and substituted into the Archimedean copula function, so the main work in this part is to calculate the probability when $x_i \ge 0$. Take computing $P(x_1 \ge 0, x_2 \ge 0, x_3 \ge 0)$ in a three-component system as an example, components in the same type with the same marginal cumulative distribution probability $F(x_i)$ and $f(x_1, x_2, x_3)$ represents the multidimensional joint probability density distribution. The calculation diagram is shown in Fig.11.

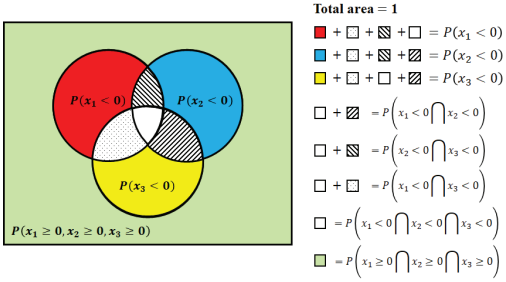

Figure 11. The probability model diagram for solving $P(x_1 \ge 0, x_2 \ge 0, x_3 \ge 0)$

It can be found from Fig. 1 that $p(x_1 \ge 0, x_2 \ge 0, x_3 \ge 0)$ can be calculated in Eq. (22):

$$P(x_1 \ge 0, x_2 \ge 0, x_3 \ge 0) = 1 - P\left(x_1 < 0 \bigcup x_2 < 0 \bigcup x_3 < 0\right). \quad (22)$$

The inclusion-exclusion principle [12] to calculate union set size is shown in Eq. (23):

$$\left|A_1 \bigcup A_2 \bigcup \dots \bigcup A_m\right| = \sum_{1 \le i \le m} |A_i| - \sum_{1 \le i < j \le m} \left|A_i \bigcap A_j\right| + \sum_{1 \le i < j < k \le m} \left|A_i \bigcap A_j \bigcap A_k\right| - \dots + (-1)^{m-1} \left|A_i \bigcap A_j \bigcap \dots \bigcap A_k\right|. \quad (23)$$

where $A_i (i = 1,2, \dots, m)$ represents the subset i; $|\cdot|$ refers to the size of set. According to the exchangeability of components, the probability between any vector state with the same number of failure components is equal. Therefore, Eq. (22) can be calculated in the following form of Eq. (24):

$$P(x_1 \ge 0, x_2 \ge 0, x_3 \ge 0) = 1 - P(x_1 < 0) - P(x_2 < 0) - P(x_3 < 0) + P\left(x_1 < 0 \bigcap x_2 < 0\right) + P\left(x_1 < 0 \bigcap x_3 < 0\right) + P\left(x_2 < 0 \bigcap x_3 < 0\right) - P\left(x_1 < 0 \bigcap x_2 < 0 \bigcap x_3 < 0\right). \quad (24)$$

It can be obtained in Eq. (25)- (27) from Sklar theorem:

$$P(x_1 < 0) = \int_{-\infty}^{0} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x_1, x_2, x_3) dx_3 dx_2 dx_1 = C(F(0), 1, 1; \varphi_\theta) = C(1, F(0), 1; \varphi_\theta) = C(1, F(0), 1; \varphi_\theta) = P(x_2 < 0) = P(x_3 < 0). \quad (25)$$

$$P\left(x_1 < 0 \bigcap x_2 < 0\right) = \int_{-\infty}^{0} \int_{-\infty}^{0} \int_{-\infty}^{\infty} f(x_1, x_2, x_3) dx_3 dx_2 dx_1 = C(F(0), F(0), 1; \varphi_\theta) = C(F(0), 1, F(0); \varphi_\theta) = C(1, F(0), F(0); \varphi_\theta) = P\left(x_1 < 0 \bigcap x_3 < 0\right) = P\left(x_3 < 0 \bigcap x_3 < 0\right). \quad (26)$$

$$P\left(x_1 < 0 \bigcap x_2 < 0 \bigcap x_3 < 0\right) = \int_{-\infty}^{0} \int_{-\infty}^{0} \int_{-\infty}^{0} f(x_1, x_2, x_3) dx_3 dx_2 dx_1 = C(F(0), F(0), F(0); \varphi_\theta). \quad (27)$$

Finally, for the three components with the same type, the calculation formula of $P(x_1 \ge 0, x_2 \ge 0, x_3 \ge 0)$ is in Eq. (28), each of which can be calculated by substituting corresponding $F(0)$ or 1 into the copula function.

$$P(x_1 \ge 0, x_2 \ge 0, x_3 \ge 0) = (-1)^0 \cdot \binom{3}{0} \cdot C(1,1,1; \varphi_\theta) + (-1)^1 \cdot \binom{3}{1} \cdot C(F(0), 1, 1; \varphi_\theta) + (-1)^2 \cdot \binom{3}{2} \cdot C(F(0), F(0), 1; \varphi_\theta) + (-1)^3 \cdot \binom{3}{3} \cdot C(F(0), F(0), F(0); \varphi_\theta). \quad (28)$$

According to the principle of inclusion-exclusion, the probability structure is further extended to the subsystem with $m_k > 3$ components and the system containing K types component. The corresponding probability structure for $P(C_k = l_k)$ and $P(\bigcap_{k=1}^{K} \{C_k = l_k\})$ are shown in Eq. (29):

$$P(C_k = l_k) = \binom{m_k}{l_k} \cdot P(x_1 \ge 0, \ \dots, x_{l_k} \ge 0, x_{l_k+1} < 0, \dots, \ x_{m_k} < 0) = \binom{m_k}{l_k} \cdot \left[\sum_{s_k=0}^{l_k} (-1)^{s_k} \cdot \binom{l_i}{s_i} \cdot C\left(\underbrace{1, \dots, 1}_{l_k - s_k}, F_k(0), \dots, F_k(0); \varphi_\theta \right) \right]. \quad (29)$$

5.4 Case study

A bridge system shown in Fig. 12 is used here to compare the reliability analysis for simulation method and proposed analytical method. Suppose component 1, 2 and 3 are in the same type 1 and component 4, 5 and 6 are in the same type 2. Here the failure time distribution is used as marginal cumulative distribution to perform reliability analysis. It is also assumed that components in the same type follow the same marginal exponential failure distribution. Two different Clayton copula functions are used to describe the internal component dependencies of each subsystem, and a third Clayton copula function is used to describe the dependencies between subsystems. The parameters are shown in Tab.3. The analytical system reliability proposed for system with dependent components

is compared with the results by simulation algorithm in [13] with 10000 samples as shown in Fig. 12.

Table 3. Failure rate of marginal exponential distributions for each type and parameters in copula.

Parameter	Definition	Value
λ_1	Failure rate of component type 1 in exponential distribution	1
λ_2	Failure rate of component type 1 in exponential distribution	1.5
θ_1	Clayton copula parameter on type 1 component	1.5
θ_2	Clayton copula parameter on type 2 component	1
θ_0	Clayton copula parameter join type 1 and type 2 copula functions	0.5

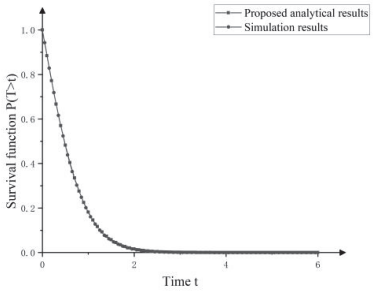

Figure 12. Comparison between proposed analytical method and simulation method

It can be seen from Fig. 11 that the analytical method of reliability for the related systems proposed in this paper is consistent with the simulation method. For the bridge structure in Fig. 10, with 10,000 simulation samples, the maximum error of the two calculation results in the first 1.5 years is 2.9939%. When the time t increases, the reliability decreases and the error between the simulation method and the analytical method increases. This is because when the reliability becomes smaller, the sampling efficiency of Monte Carlo simulation becomes lower. For example, when t=1.3 and the sampling number is 100000 rather than 10000, the reliability calculated by simulation method is 0.91440 with error 1.1472%. However, when the sample number is 10000, the running time of the simulation method for 0-6 years and the step length of 0.1 years is 2.786s, and when the sample number is 100000, the calculation time is 26.408s, while the proposed analytical method only needs 1.075s.

6. Conclusion

In this paper, a reliability approach is developed to analyse systems with independent but not identical components and an analytical reliability analysis method is proposed for systems with dependent components. In the independent but non identical situation, the divide-and-conquer algorithm is proposed to calculate the probability structure of large and complex systems. A combination of simulation method, divide and conquer algorithm and BDD structure are employed to derive the corresponding system equations of various state vectors and calculate the structure function values. The results showed the divide-and-conquer algorithm is more efficient in calculating probability structure compared to traditional exhaustive algorithm, especially when the total number of components is 1000, the calculation time can be shortened about 1000 times compared with the traditional method. In system structure calculation, the new simulation method has higher sampling efficiency, especially for the system with small failure probability and large number of components. In dependent components situation, the combination of copula theory and survival signature solve the exchangeability assumption problem. The inclusion-exclusion-principle based method could calculate probability structure effectively and results show that the proposed method is correct compared with simulation method.

References

[1] Samaniego FJ. System signatures and their applications in engineering reliability, vol.110. Springer Science & Business Media, NewYork; 2007.

[2] Geng, F. . (2018). Survival signature-based reliability approach for complex systems susceptible to common cause failures. International Journal of Reliability, Quality and Safety Engineering, 26(4).

[3] Coolen-Maturi, T. , Coolen, F. , & Balakrishnan, N. . (2020). The joint survival signature of coherent systems with shared components.

[4] Patelli, E. , Feng, G. , Coolen, F. , & Coolen-Maturi, T. . (2017). Simulation methods for system reliability using the survival signature. Reliability Engineering & System Safety, 167, 327-337.

[5] Kleinberg, & Tardos, Eva. (2014). Algorithm design (First edition..; International ed.).

[6] Efraimidis, P. , & Spirakis, P. . (2008). Weighted random sampling. Springer New York.

[7] Efraimidis, P. S. , & Spirakis, P. G. . (2006). Weighted random sampling with a reservoir. Information Processing Letters, 97(5), 181-185.

[8] Steven, J., Friedman, K. , J., & Supowit. (1987). Finding the optimal variable ordering for binary decision diagrams. 24th ACM/IEEE Design Automation Conference.

[9] Hardy, G. , Lucet, C. , & Limnios, N. . (2007). K-terminal network reliability measures with binary decision diagrams. IEEE Transactions on Reliability, 56(3), 506-515.

[10] Feng, G. , Patelli, E. , Beer, M. , & Coolen, F. . (2016). Imprecise system reliability and component importance based on survival signature. Reliability Engineering & System Safety, 150, 116-125.

[11] Nelsen R B. An Introduction to Copulas [M]. 2nd ed. New York: Springer,2006.

[12] Kenneth H. Rosen . Discrete Mathematics and Its Applications . 1221 Avenue of Americas, New York, NY 10020: McGraw-Hill Higher Education, 2002/07/08: 87

[13] Behrensdorf, J., Broggi, M., & Beer, M. (2019). Reliability Analysis of Networks Interconnected With Copulas. ASCE-ASME JOURNAL OF RISK AND UNCERTAINTY IN ENGINEERING SYSTEMS PART B-MECHANICAL ENGINEERING, 5(4). doi:10.1115/1.4044043