Advances in Reliability, Safety and Security, Part 4 – Kołowrocki, Dąbrowska (eds.)
 ISBN 978-83-68136-16-6 (printed), ISBN 978-83-68136-03-6 (electronic)

Advances in Reliability, Safety and Security

ESREL 2024
Monograph Book Series

Efficient Estimation Of Survival Signatures Through Simulation With Depth-First Search Of Indices

Sean Reed, Magnus Löfstrand

Örebro University, Örebro, Sweden

Abstract

The survival signature summarises the reliability structure of a system from which various reliability metrics can be computed. The availability of the survival signature facilitates new and more efficient types of reliability assessment for complex systems, such as those with component failure dependencies, compared to traditional approaches. However, for complex systems with many components and multiple component types, obtaining the exact signatures can be infeasible. Monte-Carlo simulation enables the computation of estimates, however obtaining accurate results can be computationally expensive. In this paper, a novel method for estimating the survival signature of general coherent systems with any number of component types through simulation is introduced. This is based on a depth-first search through a conceptualised representation of the survival signature as a directed acyclic graph, where nodes represent indices and edges represent the failure of a component of a certain type. The efficiency of the simulation is increased by eliminating redundant computation of the system state for a given component state vector based on the already known states at neighbouring indices and coherent nature of the system. On a benchmark network reliability problem, after 200000 system state evaluations, it achieved a mean absolute percentage error in the estimated signature compared to the true signature of 4.49% compared to 7.17% or greater from existing simulation methods.

Keywords: reliability, survival signature, Monte Carlo Simulation

1. Introduction

The survival signature was introduced by Coolen and Coolen-Maturi (2012) as an extension of the system signature (Boland and Samaniego, 2004) to simplify the analysis of reliability for complex coherent systems with multiple types of components that each have exchangeable failure times. The survival signature provides a summary of the reliability structure function, completely separate from the probabilistic information on component failure times. Some recent applications include reliability assessment of the hydraulic system in wind turbines (Li et al., 2020), importance analysis of uncertain systems with common cause failures (Mi et al., 2020), and reliability analysis of general phased mission systems (Huang et al., 2018). Methods for efficiently computing the exact survival signature of systems have been presented in the past, including a method for computing the survival signature from a binary decision diagram representation of the system reliability structure (Reed, 2017) (e.g. derived from a system fault tree) and for k-terminal network problems (Reed et al., 2019). However, for systems with complex reliability structures and large numbers of components and component types, exact computation is not always feasible due to the curse of dimensionality. For this reason, there is growing interest in the use of Monte Carlo simulation approaches to estimate the survival signature of systems. This involves sampling random component failure combinations, determining whether they result in system functioning or failure, and then using these results to derive statistical estimates for the signature values. However, obtaining accurate estimates in this way can also be computationally expensive due to the large number of samples required. A number of improvements to survival signature estimation through simulation have recently appeared in the literature, including the application of percolation theory to reduce the size of the signature requiring computation in network reliability problems (Behrensdorf et al., 2021), an entropy-driven method that guides the simulation towards the areas of the signature with greatest uncertainty (Di Maio et al.,

2023), and a method that transforms the signature computation for two-terminal networks with two types of components into a bi-objective optimisation problem (Lopes da Silva and Sullivan, 2023). The main contribution in this paper is a new simulation method for estimating survival signatures of general coherent systems with any number of component types based on a depth-first search (DFS) of the survival signature. The key idea is to increase the efficiency of the simulation procedure by deducing whether the system functions or fails when a given subset of the system components function, corresponding to a particular index of the survival signature, based on already computed system states corresponding to neighbouring indices.

The remainder of this paper is structured as follows: Section 2 presents the theory on the survival signature, Section 3 gives an overview of existing simulation methods for estimating the survival signature from the literature, Section 4 introduces the new simulation method based on a DFS of the signature indices, Section 5 describes an example network reliability problem used to test the new method, Section 6 presents results and analysis comparing the relative accuracy the new method to existing methods when applied to the example reliability problem, and Section 7 gives some final conclusions.

2. Survival signature

Let $x = (x_1, x_2, \dots, x_m) \in \{0,1\}^m$ represent a Boolean state vector for a system of m components with exchangeable failure times, where $x_i = 1$ if component i functions and $x_i = 0$ if it is failed. Also let $\phi: \{0,1\}^m \rightarrow \{0,1\}$ represent the system reliability structure function, defined for all 2^m possible x, where $\phi(x) = 1$ if the system functions with component states x and $\phi(x) = 0$ if it is failed. In this paper, only coherent systems will be considered, defined mathematically as systems where $\phi(x)$ is not decreasing in any of the components of x, meaning that a component failure cannot cause an otherwise failed system to function. Let S_l denote the set of component state vectors with exactly l of the m components functioning (i.e. $\sum_{i=1}^{m} x_i = l$). The survival signature is then defined as the vector Φ where the value at index $l \in \{0,1,2, \dots, m\}$, denoted Φ_l, gives the probability that the system functions given that precisely l components function:

$$\Phi_l = \binom{m}{l}^{-1} \sum_{x \in S_l} \phi(x) \tag{1}$$

Now consider the case where the m components in the system are partitioned into K different types, where the M_k components of type $k \in \{1, \dots, K\}$ have exchangeable random failure times. Let $S_{l_1,\dots,l_K}$ denote the set of component state vectors that contain precisely $l_k \in \{0,1, \dots, M_k\}$ functioning components of type k (i.e. those for which $\sum_{i=1}^{M_k} x_i^k = l_k$ for $k = 0,1, \dots, K-1$ where x_i^k is the i*th* component of type k). Following the notation from Behrensdorf et al. (2021), $\underline{l}$ will be used as shorthand for $l_1, \dots, l_K$ in the remainder of this paper. Also let $\left|S_{\underline{l}}\right| = \prod_{k=1}^{K} \binom{M_k}{l_k}$ denote the cardinality of $S_{\underline{l}}$ and $\overline{\Phi}_{\underline{l}} = \sum_{x \in S_{\underline{l}}} \phi(x)$ denote the number of state vectors from $S_{\underline{l}}$ for which the system functions. The generalised survival signature, Φ, is then defined as the multidimensional array with K dimensions where the value at index $\underline{l}$, denoted $\Phi_{\underline{l}}$, gives the probability that the system functions if precisely $(l_1, \dots, l_K)$ components of types $(1, \dots, K)$ respectively function:

$$\Phi_{\underline{l}} = \frac{\overline{\Phi}_{\underline{l}}}{\left|S_{\underline{l}}\right|} \tag{2}$$

In total, the survival signature for a system has Ω unique indices:

$$\Omega = \prod_{k=1}^{K} (M_k + 1) \tag{3}$$

The survival signature for a coherent system is a non-decreasing function in each dimension of its index, since the probability of the system functioning may increase, but can never decrease, as the number of components of each type that function increases:

$$\Phi_{\underline{l^a}} \leq \Phi_{\underline{l^b}} \; if \; l_k^a \leq l_k^b, \forall k \in \{1, \dots, K\},$$

where $\underline{l^a}$ and $\underline{l^b}$ are any two indices of the surivival signature. Ignoring the degenerate cases of systems that are always functioning or always failed, it follows that:

$$\Phi_{\underline{1}} = 1,$$
$$\Phi_{\underline{0}} = 0,$$

where $\underline{1}$ is the index representing the case that all components function and $\underline{0}$ representing the case that all components fail.

3. Existing simulation methods for estimating survival signature

In this section, the general procedure for estimating survival signatures of a system is described and a review of enhanced methods from the literature is presented. The simplest simulation method, referred to herein as the naïve method, is to approximate the survival signature value at each index $\underline{l}$ by repeating simulation trials, where each trial consists of sampling a random component state vector $x \epsilon S_{\underline{l}}$, and then evaluating whether this results in a system functioning or system failed state. Typically, a random component state vector $x \epsilon S_{\underline{l}}$ is generated by sampling a random permutation for the components among each type $k \in \{1, \dots, K\}$ and then setting the first l_k components of each type in the permutation to functioning and the remaining components to failed. The survival signature value at index $\underline{l}$, denoted $\widehat{\Phi}_{\underline{l}}$, is then estimated as the proportion of the $n_{\underline{l}}$ trials evaluated at that index that resulted in the system functioning:

$$\widehat{\Phi}_{\underline{l}} = \frac{\alpha_{\underline{l}}}{\alpha_{\underline{l}} + \beta_{\underline{l}}} = \frac{\alpha_{\underline{l}}}{n_{\underline{l}}} \tag{4}$$

where $\alpha_{\underline{l}}$ and $\beta_{\underline{l}}$ are the number of trials at index $\underline{l}$ that resulted in the system functioning and failing, respectively.

The equivalence between the estimated signature from (4) and the true signature from (2) can be clearly seen, with the former approximating the full set of component state vectors in $S_{\underline{l}}$ by random draws from $S_{\underline{l}}$ with replacement. Equation (4) is the maximum likelihood estimator and will converge toward the true value as the number of simulation trials is increased. The main computational expense is from evaluating the system state for a given component state vector through the system reliability model. Typically, a fixed number of simulation trials are performed at each index or until some stopping point is reached, such as when the coefficient of variation, given by the equation below, falls to a predetermined threshold value (Behrensdorf et al., 2021):

$$CV_{\underline{l}} = \frac{\left(\widehat{\Phi}_{\underline{l}} - \widehat{\Phi}_{\underline{l}}^{\,2}\right)/n_{\underline{l}}}{\widehat{\Phi}_{\underline{l}}} \tag{5}$$

The number of indices in the survival signature is $\prod_{k=1}^{K}(M_k + 1)$ and can therefore be very large, particularly for systems with many components and component types. The total number of random component state vectors that must be generated and number of system state evaluations required by the naïve method to converge to an accurate survival signature estimate can therefore be huge. Recent research has therefore focused on improving the efficiency of convergence compared to the naïve method, either by exploiting knowledge of the system reliability problem or optimising the sampling scheme.

Behrensdorf et al. (2021) presented a method for estimating the survival signatures of large networks with unreliable nodes using percolation theory. According to percolation theory, when the proportion of failed nodes reaches a critical threshold f_c that depends only on the network structure, specifically the first and second moments of its degree distribution, there is only negligible probability that the network functions. Therefore, the value of the survival signature at any index $\underline{l}$ where $\frac{1}{m}\sum_{k=1}^{K} l_k$ is less than $(1 - f_c) \times m$ can be set to 0 without further analysis and only the remaining indices need to be estimated through simulation. The computational effort can therefore be reduced significantly compared to the naïve method for networks with a low critical threshold. However, the method is not applicable to systems other than networks and nor does it offer improved efficiency in evaluating the remaining indices. Di Maio et al. (2023) developed an entropy-driven method based on a Bayesian framework and information theory to more efficiently utilise a limited computational budget when obtaining an estimated survival signature through simulation. This method, which is applicable to general systems, follows the naïve method except that before each simulation trial, the value of the survival signature at each index $\underline{l}$ is assumed to be described by a prior Beta Distribution with parameters $\alpha_{\underline{l}}$ and $\beta_{\underline{l}}$, and the index to evaluate next is then chosen such that the expected information gain is maximised. The expected information gain from performing the next trial at a certain index is determined as the weighted sum of the information gains if the trial were to result in system functioning or failure, where the weights are given by the prior estimates of the survival signature at that index. The posterior distribution at the evaluated index is then calculated by incrementing $\alpha_{\underline{l}}$ or $\beta_{\underline{l}}$, depending on whether the system functioned or failed in the simulation trial respectively, and the process continues. They combined this with the percolation theory method from Behrensdorf et al.

(2021) to reduce the number of indices that require simulation when analysing network problems. The method therefore improves the overall efficiency of finding the signature by prioritising evaluation at regions of the signature where gains are expected to be greatest. The methods from Behrensdorf et al. (2021) and Di Maio et al. (2023) therefore differ from the naïve method in excluding indices from evaluation where the network will almost certainly be failed and guiding the order in which indices are simulated. However, each update at an index requires sampling a random component state vector and then evaluating the system state that results. These are typically computationally expensive operations, particularly the evaluation of the system state. Note that for coherent systems, the estimated survival signature from the naïve method, or the improved methods from Behrensdorf et al. (2021) and Di Maio et al. (2023), is also not guaranteed to be a non-decreasing function in each dimension of its index. This is because the value of the survival signature is approximated independently at each index $\underline{l}$ as the proportion of the randomly sampled (with replacement) state vectors from $S_{\underline{l}}$ that resulted in the system functioning. Therefore, if the value at index $\boldsymbol{a}$ is underestimated or the value at index $\boldsymbol{b}$ is overestimated, then it is possible for $\widehat{\Phi_{\underline{l}^a}} > \widehat{\Phi_{\underline{l}^b}}$ even if $l_k^a \leq l_k^b, \forall k \in \{1, \dots, K\}$, where $\underline{l^a}$ and $\underline{l^b}$ are two indices of the survival signature. Certain types of analysis, such as the importance measure analysis approach by Rusnak et. al. (2024), assume a non-decreasing signature function making the use of estimates derived from such methods potentially unsuitable.

Lopes da Silva and Sullivan (2023) showed how simulation and bi-objective optimisation could be combined to efficiently estimate the survival signature of two-terminal networks with unreliable nodes of two types. In each simulation trial, as in the naïve method, a permutation of the order for component failures is first randomly sampled for each component type. Two sets of capacities are then assigned to the nodes in the network, one for each of the component types. For each component type, the capacity of a node is set to its order among the failure time of components of that node type if it is of the same type, or to infinity if the node is of the other type. By then solving the bi-objective maximum capacity path problem between the terminal nodes, using a modified bi-objective Dijkstra algorithm (Sedeño-noda and Colebrook, 2019) based on the node capacities for the two component types, the non-dominated set of solutions is found, where each solution comprises the maximal flows along a path between the terminal nodes for the two sets of node capacity assignments for the network. Since these solutions correspond to the minimum number of components of each type that fail in the simulated permutations before the system fails, the system state at every index of the survival signature can be determined in each simulation trial. The method also ensures that the estimated survival signature is a non-decreasing function, however it can only be applied to two-terminal networks with two types of component which limits its applications.

4. Depth-first search simulation method for estimating survival signature

A new method for estimating the survival signature of a coherent system through simulation is introduced in this section that is applicable to general systems (including, but not limited to, networks) with any number of component types, eliminates redundant computations of the system state to increase computational efficiency, and results in a non-decreasing survival signature estimate. As with all simulation methods, it requires a system reliability model from which the system functioning or failed state can be determined for a given combination of component functioning or failed states. Specifically, the functions shown in Table 1 are assumed to be available.

Table 1. Required functions of the system reliability model.

Function name	Arguments	Description
get_num_types	None	Returns the number of component types in the system, i.e. K.
get_num_components	k	Returns the number of components of type k in the system, i.e. M_k.
get_components	k	Returns the components of type k in the system.
set_component_failed	i	Sets component $i \in \{1,\dots,m\}$ to the failed state.
set_component_functioning	i	Sets component $i \in \{1,\dots,m\}$ to the functioning state.
get_state	None	Returns true if the system is in a functioning state given the current state of the system components, otherwise returns false.

In general, calling the *get_state()* function from Table 1 on a reliability model is computationally expensive. Therefore, the aim is to approximate the survival signature for the system as accurately as possible within a given budget that defines the number of times this function can be called. The simulation algorithm only interacts with the system reliability model through these functions, without requiring further knowledge of its composition or implementation, thus viewing it as a "black-box". It can therefore be applied to any type of system reliability model such as networks, fault trees, or Markov models, and is able to analyse systems

containing any number of component types. For the purposes of the new method, the survival signature is viewed conceptually as a directed acyclic graph, where each node:

- Represents and is labelled with a unique index $(l_1, \dots, l_K)$ from the signature.
- Has at most K parent nodes representing signature indices with exactly one more functioning component. Each parent is connected by an incoming edge with a different label $k \in \{1, \dots, K\}$ matching the type of the additional surviving component and represents index $(l_1, \dots, l_{k-1}, l_k + 1, l_{k+1}, \dots, l_K)$ of the signature. The shorthand $\underline{l}^{+k}$ will be used in the remainder of this paper to refer to the index $(l_1, \dots, l_{k-1}, l_k + 1, l_{k+1}, \dots, l_K)$ relative to index $\underline{l}$.
- Has at most K child nodes representing signature indices with exactly one more failed component. Each child is connected by an outgoing edge with a different label $k \in \{1, \dots, K\}$ matching the type of that additional failed component and represents index $(l_1, \dots, l_{k-1}, l_k - 1, l_{k+1}, \dots, l_K)$ of the signature. The shorthand $\underline{l}^{-k}$ will be used in the remainder of this paper to refer to the index $(l_1, \dots, l_{k-1}, l_k - 1, l_{k+1}, \dots, l_K)$ relative to index $\underline{l}$.

The graph has a single source node (i.e. node without parent nodes) with K child nodes representing index $(M_1, \dots, M_K)$ of the signature (i.e. all components in the system function), and a single terminal node (i.e. node without child nodes) with K parent nodes, representing index (0, ... ,0) of the signature (i.e. all components in the system are failed). An example of part of the directed acyclic graph representation of a survival signature is shown in Fig. 1.

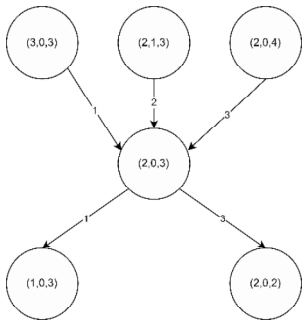

Fig. 1. An example of part of a directed acyclic graph of a survival signature showing the node representing index (2,0,3) and its parent and child nodes.

Consider the case where a permutation for the components of each type in the system is generated and the system state is evaluated at each index under the assumption that the first l_k components in the permutation for type *k* function and the remaining components fail. For a coherent system, it follows that if a node in the directed acyclic graph representation of the survival signature corresponds to a system failed state, its child nodes will also correspond to system failed states. For example, indices (1,0,3) and (2,0,2) from Fig. 1 must represent system failed states if index (2,0,3) represents a system failed state. This is because the child node connected by an edge labelled *k* is evaluated with a component state vector containing the same set of failed components as its parent, but with an additional failed component of type *k*. Similarly, if a node corresponds to a system working state, each of its parent nodes will also correspond to system working state since it is evaluated with a component state vector containing the same set of surviving components except for an additional surviving component of type *k* that matches the label of the edge between them. For example, if at least one of indices (1,0,3) and (2,0,2) from Fig. 1 represent a system functioning state, then index (2,0,3) must also represent a system functioning state. A key idea of the DFS-based simulation method is to exploit this relationship between the system states of connected nodes in the directed acyclic graph to eliminate redundant computations of the system functioning or failed state at each index.

An overview of the main steps in the algorithm for the new method is given by the flowchart in Fig. 2. In each iteration of the algorithm, provided sufficient budget for system reliability model evaluations remains, a DFS of the survival signature graph is performed to update the system functioning and failed counts from (4) at every index based on a new random permutation of the components of each type. Once the budget is expended, the survival signature is estimated from (4). A DFS starts at the source node, representing index $(M_1, \dots, M_K)$, and explores as far along each branch toward the terminal node, representing index (0, ... ,0), as possible before backtracking. When each node is visited, the system state corresponding to the component state vector with the first l_k components in the permutation for each component type k functioning, and the remaining components

failed, is determined. The number of failed components is therefore incremented by one of type k compared to the component state vector evaluated at its parent node. First, an attempt is made to determine this by checking the system state represented by the parent node and, if necessary, checking the system state represented by any child nodes that have already been visited. Only if the system state cannot be determined from the connected nodes, is it determined by evaluating the system reliability model with the component state vector. Due to the incremental increase in component failures during the DFS, the computational expense of the system state evaluation may also be reduced for many system reliability models. A record is kept of already visited nodes to ensure each node in the graph of indices is visited exactly once in each iteration of the simulation.

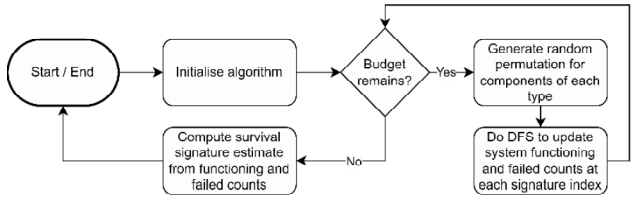

Fig. 2. Flowchart describing the main steps in the DFS Survival Signature Simulation Algorithm.

```
def get_dfs_signature_estimate(system, budget):
    # Create arrays to store survived and failed counts, each having K dimensions and
    # length M_k + 1 in dimension k.
    num_types = system.get_num_types()
    array_shape = (system.get_num_components(k) + 1 for k ϵ {1, …, num_types})
    survived = array of shape array_shape where all entries are 0.
    failed = array of shape array_shape where all entries are 0.

    # Perform DFS simulations of survival signature graph until budget expended.
    total_evaluations = 0
    while total_evaluations < budget:
        remaining_budget = budget - total_evaluations
        total_evaluations += simulate_dfs(system, survived, failed, remaining_budget)

    # Perform element-wise division and addition to obtain signature estimate from counts.
    signature = survived / (survived + failed)

    return signature
```

Fig. 3. Pseudo-code for the function get_dfs_signature_estimate. Calls function simulate_dfs given in Fig. 4.

```
def simulate_dfs(system, survived, failed, budget):

    # Create random permutation of components for each type. component_permutation[k][i] is the ith
    # component in the permutation of components of type k.
    num_types = system_model.get_num_types()
    component_permutations = []
    for k ϵ {1, …, num_types}:
        type_k_components = system_model.get_components(k)
        component_permutations.append(random permutation of type_k_components)

    # Create initial index l representing all components functioning.
    l = (system_model.get_num_components(k) for k ϵ {1, …, num_types})

    # Create visited and system state arrays, where all entries are False.
    array_shape = (system.get_num_components(k) + 1 for k ϵ {1, …, num_types})
    visited = array of shape array_shape where all entries are False.
    system_states = array of shape array_shape where all entries are False.

    # Begin depth-first search from l.
    num_evaluations = dfs_from_l(system, l, True, component_permutations, visited,
                                 system_states, survived, failed, budget)
    return num_evaluations
```

Fig. 4. Pseudo-code for the function simulate_dfs. Calls function dfs_from_l given in Fig. 5.

The pseudo-code for the function named get_dfs_signature_estimate that performs the initialisation, initiates the simulation, and calculates the survival signature estimate for a system, is given in Fig. 3. The inputs to this function are the parameter *system*, which represents the system reliability model with the functions described in

Table 1, and the parameter ***budget***, which is an integer value representing the number of calls that can be made to the function ***get_state*** on the system reliability model during computation of the estimate. Note that it is assumed that the system reliability model is initially in the state where all components are functioning. The get_dfs_signature_estimate function performs repeated DFS of the survival signature by calling the function simulate_dfs, for which the pseudo-code is given in Fig. 4. To initiate a DFS, simulate_dfs calls the function dfs_from_l, for which the pseudo-code is given in Fig. 5, to begin the DFS from the source node, representing all components functioning, which then calls itself recursively to visit each non-visited child node to extend the DFS along each branch.

```
def dfs_from_l(system, l, parent_survived, component_permutations, visited, system_states,
               survived, failed, budget):
    visited[l] = True
    num_evaluations = 0

    # Generate indices of children to visit in random order.
    num_types = system_model.get_num_types()
    child_indices = []
    for k ϵ {1, … , num_types}:
        if l[k] != 0:
            child_index = (l[i] - 1 if i == k else l[i] for  iϵ {1, … , num_types})
            child_indices.append((k, child_index))
    shuffle child_indices into random permutation

    # Determine state of system at this index.
    if parent_survived:
        # Check if this index has any visited child that survived.
        visited_surviving_child = False
        for k, child_index in child_indices:
            if visited[child_index] and system_states[child_index]:
                visited_surviving_child = True
                break loop
        if visited_surviving_child:
            state = True
        else:
            state = system.get_state()
            num_evaluations += 1
    else:  # If system failed at parent index, will also be failed at this index.
        state = False

    # Update arrays with state of system at this index.
    if state:
        survived[l] += 1
    else:
        failed[l] += 1
    system_states[l] = state

    # Continue depth-first search along child node branches.
    for k, child_index in child_indices:
        if parent_survived and num_evaluations >= budget:
            break loop
        if not visited[child_index]:
            if parent_survived:
                # Set additional component to failed for child node.
                component_to_fail = component_permutations[k][child_index[k]]
                system.set_component_failed(component_to_fail)
            # Continue depth-first search from child node.
            remaining_evaluations = budget - num_evaluations
            num_evaluations += dfs_from_l(system, child_index, state, component_permutations,
                 visited, system_states, survived, failed, remaining_evaluations)

            if parent_survived:
                # Restore component to functioning that was set to failed for child node.
                system.set_component_functioning(component_to_fail)

    return num_evaluations
```

Fig. 5. Pseudo-code for the function dfs_from_l.

5. Example Network Reliability Problem

To analyse the performance of the DFS-based simulation method, and compare it to the existing alternatives, a network reliability problem was used. This problem was originally adapted from a model of the electricity transmission network of Great Britain by Behrensdorf et al. (Behrensdorf et al., 2021) and has previously been used to analyse the survival signature estimation methods by Behrensdorf et al. (Behrensdorf et al., 2021) and Di Maio et. al. (Di Maio et al., 2023). The network consists of 29 nodes, each representing an unreliable component in the system, that are split into two component types, connected by perfectly reliable edges, as shown in Fig. 6.

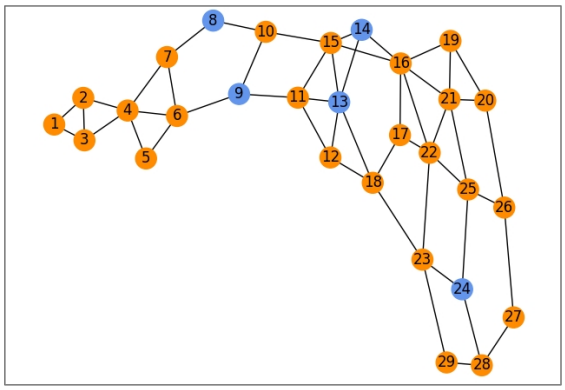

Fig. 6. Topology of a network based on the electricity transmission network of Great Britain with 29 nodes representing unreliable components, where the nodes labelled 8, 9, 13, 14, and 24 are components of type 1 and the remainder are components of type 2.

It is assumed that the efficiency of network G, comprising n nodes, is defined by:

$$E(G) = \frac{1}{n(n-1)} \sum_{i \neq j \in G} \frac{1}{d(i,j)} \tag{6}$$

where and $d(i,j)$ is the length of the shortest path between nodes i and j through only functioning nodes.

The network is deemed to be failed if its relative efficiency falls to below 50% of its nominal efficiency:

$$\psi(x) = \frac{E\big(G(x)\big)}{E(G)} < 0.5, \tag{7}$$

where $G(x)$ is the network under component state vector x and G is the network when all nodes function.

The survival signature for this system has 150 indices and the exact signature value for each has been calculated using the brute-force and computationally expensive approach of evaluating the system state for every possible component state vector.

6. Results for example network reliability problem

The new DFS-based simulation method introduced in this paper, the naïve simulation algorithm, the percolation-based from Behrensdorf et al. (Behrensdorf et al., 2021) and entropy-based simulation algorithms with percolation from Di Maio et. al. (Di Maio et al., 2023) were each applied to estimating the survival signature for the network reliability problem described in Section 5. It was not possible to apply the bi-objective optimisation simulation algorithm from Lopes da Silva and Sullivan (Lopes da Silva and Sullivan, 2023) to this problem, since it is only applicable to two terminal network problems with two types of components. Each algorithm was given a budget of 200000 system state evaluations, except for the entropy-based algorithm which was given a budget of 205000 including evaluations used during initialisation, to estimate the signature. The entropy-based algorithm requires initial simulation of each index to calculate its initial entropy value. For this purpose, each index was simulated 100 times following the naïve method, requiring a total of 15000 system state evaluations across all 150 indices, as performed previously by Di Maio et. al. (Di Maio et al., 2023) when analysing this network problem. Since simulation is a random process, the simulations were repeated 10 times for each algorithm. The mean square errors in the estimated survival signature values compared to the true

values, across all 150 indices, were calculated after every 10000 system state evaluations. The mean values for the mean square errors from each algorithm across the 10 simulation repetitions are plotted against the number of system state evaluations in Fig. 7. This shows that the DFS-based algorithm introduced in this paper performs well on this problem, resulting in a significantly smaller mean square error than the alternative algorithms across the full range of system state evaluations.

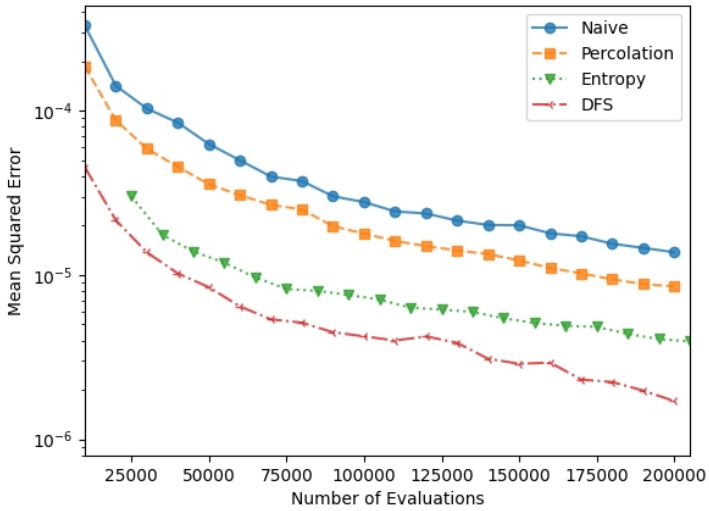

Fig. 7. Plot of the mean squared error error in estimated survival signature values across all indices against the number of evaluations of the system state for different simulation algorithms applied to the network problem described in Section 5. Mean values estimated from 10 simulation repetitions. Note the use of a logarithmic scale on the y-axis.

In Table 2, the mean absolute error and mean absolute percentage error in the survival signature estimates from each method after 200000 evaluations of the system state (205000 evaluations for the entropy-based method) are given. Both the mean values and bounds for the 95% confidence interval from the 10 simulation repetitions for each algorithm are reported. The mean absolute error for the DFS-based simulation method was 0.0004 compared to 0.0007 for the next best method (entropy-based) and the mean absolute percentage error was 4.49% compared to 7.17% for the next best method (percolation). These results suggest that the DFS-based simulation method has better convergence to the true survival signature for the network reliability problem from Section 5.

Table 2. Comparison of mean absolute and percentage error in estimated survival signature values across all indices after 200000 evaluations of the system state (205000 evaluations for the entropy-based method) between different simulation algorithms applied to the network problem described in Section 5. Mean values and 95% confidence interval (CI) bounds for the true means were calculated from 10 simulation repetitions.

Simulation Method	Mean Absolute Error (95% CI Bounds)	Mean Absolute Percentage Error (95% CI Bounds)
Naïve	0.0012 (0.0010, 0.0013)	7.43% (5.84%, 9.01%)
Percolation	0.0009 (0.0080, 0.0011)	7.17% (5.70%, 8.64%)
Entropy-based	0.0007 (0.0006, 0.0007)	7.45% (7.12%, 7.79%)
DFS- based	0.0004 (0.0004, 0.0005)	4.49% (3.86%, 5.12%)

Analysis of the DFS-based method showed that 87% of the system state evaluations used to compute the signature estimates for the example problem were deduced from the state at neighbouring indices, through the process described in Section 4, representing a significant improvement in computational efficiency compared to the naïve method.

7. Summary and conclusions

A new simulation method for estimating the survival signature of a coherent system based on a DFS of the indices has been presented. The method exploits the fact that the signature is a non-decreasing function in each dimension of its index to, where possible, deduce the system state at an index based on known system states at

neighbouring indices. It also has the advantage that many system evaluations are incremental, by advancing component failures one at a time, reducing the average computational expense of evaluating the system state for many reliability problems. Results from comparing the mean error in the estimated survival signature with existing simulation methods on a benchmark network reliability problem, presented in Table 2, showed it was able to achieve significantly improved accuracy for a given computational budget of system evaluations. The new method also has the advantage of guaranteeing that the survival signature estimate will be non-decreasing function in each dimension of its index. Since the new method is generally applicable all types of coherent system reliability problems where survival signatures are of interest, including those with more than two types of components, an area for further research is to benchmark its accuracy across a wider set of problem types. It is expected that the increase in efficiency, compared to the alternative simulation methods, will increase with increasing numbers of component types since the number of neighbouring indices to each index will increase. For network-based reliability problems, the DFS-based method could also be combined with percolation theory by truncated the DFS along any branch when an index representing the critical fraction of component failures is reached.

Acknowledgements

This work is carried out within the project *A general digital twin driving mining innovation through statistical and logical modelling*, funded by VINNOVA, through Swedish Mining Innovation. We gratefully acknowledge the support and funding.

References

Behrensdorf, J., Regenhardt, T.-E., Broggi, M., Beer, M. 2021. Numerically efficient computation of the survival signature for the reliability analysis of large networks. Reliab. Eng. Syst. Saf. 216, 107935. https://doi.org/10.1016/j.ress.2021.107935

Boland, P.J., Samaniego, F.J. 2004. The Signature of a Coherent System and Its Applications in Reliability, in: Soyer, R., Mazzuchi, T.A., Singpurwalla, N.D. (Eds.), Mathematical Reliability: An Expository Perspective. Springer US, Boston, MA, MA, 3-30. https://doi.org/10.1007/978-1-4419-9021-1_1

Coolen, F.P.A., Coolen-Maturi, T. 2012. Generalizing the Signature to Systems with Multiple Types of Components, in: Complex Systems and Dependability. Springer Berlin Heidelberg, pp. 115–130. https://doi.org/10.1007/978-3-642-30662-4_8

Di Maio, F., Pettorossi, C., Zio, E. 2023. Entropy-driven Monte Carlo simulation method for approximating the survival signature of complex infrastructures. Reliab. Eng. Syst. Saf. 231, 108982. https://doi.org/10.1016/j.ress.2022.108982

Huang, X., Aslett, L.J.M., Coolen, F.P.A. 2018. Reliability analysis of general phased mission systems with a new survival signature. ArXiv E-Prints. https://doi.org/10.48550/arXiv.1807.09616

Li, Y., Coolen, F.P.A., Zhu, C., Tan, J. 2020. Reliability assessment of the hydraulic system of wind turbines based on load-sharing using survival signature. Renew. Energy 153, 766-776.

Lopes da Silva, D.B., Sullivan, K.M. 2023. An Optimization-Based Monte Carlo Method for Estimating the Two-Terminal Survival Signature of Networks with Two Component Classes.

Mi, J., Beer, M., Li, Y.-F., Broggi, M., Cheng, Y. 2020. Reliability and importance analysis of uncertain system with common cause failures based on survival signature. Reliab. Eng. Syst. Saf. 201.

Reed, S., 2017. An efficient algorithm for exact computation of system and survival signatures using binary decision diagrams. Reliab. Eng. Syst. Saf. 165, 257-267. https://doi.org/10.1016/j.ress.2017.03.036

Reed, S., Löfstrand, M., Andrews, J. 2019. An efficient algorithm for computing exact system and survival signatures of K-terminal network reliability. Reliab. Eng. Syst. Saf. 185, 429-439. https://doi.org/10.1016/j.ress.2019.01.011

Rusnak, P., Zaitseva, E., Levashenko, V., Bolvashenkov, I., Kammermann, J. 2024. Importance analysis of a system based on survival signature by structural importance measures. Reliab. Eng. Syst. Saf. 243, 109814. https://doi.org/10.1016/j.ress.2023.109814

Sedeño-Noda, A., Colebrook, M. 2019. A biobjective Dijkstra algorithm. Eur. J. Oper. Res. 276, 106-118. https://doi.org/10.1016/j.ejor.2019.01.007