An Optimization-Based Monte Carlo Method for Estimating the Two-Terminal Survival Signature of Networks with Two Component Classes

Daniel B. Lopes da Silva[1] and Kelly M. Sullivan[2,*]

[1]Department of Industrial Engineering, Clemson University, Clemson, SC, USA, 29634

[2]Department of Industrial Engineering, University of Arkansas, Fayetteville, AR, USA, 72701

[*]Corresponding author: Kelly M. Sullivan, ksulliv@uark.edu

July 12, 2024

Abstract

Evaluating two-terminal network reliability is a classical problem with numerous applications. Because this problem is #P-Complete, practical studies involving large systems commonly resort to approximating or estimating system reliability rather than evaluating it exactly. Researchers have characterized signatures, such as the destruction spectrum and survival signature, which summarize the system's structure and give rise to procedures for evaluating or approximating network reliability. These procedures are advantageous if the signature can be computed efficiently; however, computing the signature is challenging for complex systems. With this motivation, we consider the use of Monte Carlo (MC) simulation to estimate the survival signature of a two-terminal network in which there are two classes of i.i.d. components. In this setting, we prove that each MC replication to estimate the signature of a multi-class system entails solving a multi-objective maximum capacity path problem. For the case of two classes of components, we adapt a Dijkstra's-like bi-objective shortest path algorithm from the literature for the purpose of solving the resulting bi-objective maximum capacity path problem. We perform computational experiments to compare our method's efficiency against intuitive benchmark approaches. Our computational results demonstrate that the bi-objective optimization approach consistently outperforms the benchmark approaches, thereby enabling a larger number of MC replications and improved accuracy of the reliability estimation. Furthermore, the efficiency gains versus benchmark approaches appear to become more significant as the network increases in size.

Keywords: Survival signature, two-terminal reliability, network reliability, Monte Carlo simulation, bi-objective optimization.

1 Introduction

In a network where some elements fail independently of the other components according to a known probability, the *two-terminal reliability* is the probability that there is at least one functional path between two specified terminal nodes s and t. Evaluating the two-terminal (and the more general K-terminal) reliability is a classical problem with applications in wired and wireless communication networks, electronic circuit design, computer networks, and electrical power distribution, among other systems (Cook and Ramirez-Marquez, 2007; Gebre and Ramirez-Marquez, 2007; Silva et al., 2015; Chakraborty et al., 2020). The problem is known to be #P-Complete in general (Valiant, 1979; Provan and Ball, 1983; Ball, 1986), and numerous exact and approximate methods have been proposed to solve it. With the emergence of massive networked structures with thousands of components, efficient algorithms for evaluating two-terminal network reliability remain an important research topic.

One attractive approach for evaluating or comparing the reliability of complex systems is to use signatures such as the system signature (Samaniego, 1985), the destruction spectrum (D-spectrum) (Gertsbakh and Shpungin, 2009), and the survival signature (Coolen and Coolen-Maturi, 2012). Loosely speaking, a *signature* is a compact summary of the system's structure (e.g., the network's topology) that, if computed, enables efficient evaluation or approximation of system reliability (e.g., the probability that s and t are connected). Whereas the system signature requires assuming components are independent and identically distributed (i.i.d.), the survival signature can be applied in cases where there are multiple classes of components and components in different classes are permitted to be non-identical.

Because computing signatures for network systems is itself computationally challenging, it is common to estimate a signature using Monte Carlo (MC) and then use the MC signature estimate to produce an estimate of network reliability. In this context, such a combined MC/signature approach is known to be advantageous compared to estimating the system reliability directly using *crude* MC (i.e., in which the system's state is repeatedly evaluated after sampling each component's time to failure). For instance, combined MC/signature approaches have been shown to have bounded relative error (Gertsbakh et al., 2016) whereas crude MC may have unbounded relative error (Elperin et al., 1991). Nonetheless, computing signatures poses a major challenge in terms of computational complexity, especially when considering large, heterogeneous networks.

In this paper, we consider the problem of estimating the two-terminal survival signature by MC simulation for a system with two classes of unreliable nodes. The contributions are as follows:

1. We show that each MC replication entails solving a multi-objective maximum capacity path problem. To the best of our knowledge, this is the first work to point out the relationship between survival signature computation and a multi-objective optimization problem.

2. We adapt the Dijkstra's-like algorithm of Sedeño-Noda and Colebrook (2019) to solve the resulting bi-objective maximum capacity path problem for a system with two classes of components.

3. We show through numerical experiments that (i) the bi-objective optimization approach consistently outperforms benchmark approaches, thereby increasing the rate at which MC replications can be performed and improving the accuracy of reliability estimation; and (ii) the efficiency gains become more significant as the network increases in size.

The remainder of this paper is organized as follows. In Section 2 we summarize literature related to the two-terminal reliability problem and algorithms proposed to solve it. In Section 3, we present an MC framework for estimating the two-terminal survival signature of a network system with two component classes and describe methods for completing the calculations within a MC replication. We discuss computational results in Section 4 and conclude in Section 5.

2 Background and Literature Review

2.1 Evaluating Two-Terminal Network Reliability

Many exact approaches to solve the two-terminal reliability problem have been proposed since the late 1950s. We refer the reader to Moore and Shannon (1956) and Barlow and Proschan (1965), for a summary of early research in this area and to Brown et al. (2020) for a general view of more recent results. Although exact algorithms have proven effective for small networks or networks with special structures, such as trees or series-parallel, approximation algorithms are more commonly used for large, generally structured networks.

The most common approaches used to evaluate two-terminal reliability (and its generalizations, e.g., K-terminal reliability and multi-state two-terminal reliability) exactly are based on methods such as sum of disjoint products (Jane and Yuan, 2001; Datta and Goyal, 2017), state-space decomposition (Aven, 1985; Alexopoulos, 1995; Bai et al., 2018), cut/path-based state enumerations (Ramirez-Marquez et al., 2006; Gebre and Ramirez-Marquez, 2007), factoring (Moskowitz, 1958; Satyanarayana and Chang, 1983; Wood, 1985, 1986; Burgos and Amoza, 2016), and binary decision diagrams (BDD) (Hardy et al., 2007; Kuo et al., 2007). The exponential nature of these algorithms has prompted research into computationally efficient bounds (Jane et al., 2009; Lê et al., 2013; Sebastio et al., 2014; Silva et al., 2015) and other methods for estimating or approximating two-terminal reliability based on neural networks (Srivaree-ratana et al., 2002; Altiparmak et al., 2009), network reduction procedures (Zhang and Shao, 2018), the cross-entropy method (Hui et al., 2005), failure frequency approximation (Heidarzadeh et al., 2018), and spline interpolation (Cristescu and Dragoi, 2021).

MC simulation has been widely utilized for estimating two-terminal reliability since the 1980s. Fishman (1986) evaluated four MC sampling methods to estimate two-terminal reliability: (1) dagger sampling, which relies on inducing negative correlations between the replications' outcome; (2) sequential destruction/construction based on permutations of the network elements; (3) bounds on the reliability; and (4) failure sets enumeration, pointing out each method's advantages and pitfalls.

Ramirez-Marquez and Gebre (2007) present an MC method to approximate bounds on the two-terminal reliability of capacitated networks based on simulating network configurations and employing classification trees to obtain minimal cut or path vectors.

Stern et al. (2017) combines MC simulation and machine learning techniques to approximate two-terminal reliability. Their method relies on MC simulation to generate components' failure samples, which are used to estimate the two-terminal reliability.

In recent years, researchers have combined MC simulation principles with the concept of a system signature to create improved methods for estimating the two-terminal reliability of complex systems. Two such methods are presented in Reed et al. (2019) and Behrensdorf et al. (2021). We discuss these methods in more details later in this paper, but first we introduce the concept of a system signature.

2.2 Signatures

To formally introduce the concept of signatures, consider a system in n components and let $\mathbf{x}$ be the state vector whose elements are

$$x_i = \begin{cases} 1, & \text{if component } i \text{ operates,} \\ 0, & \text{if component } i \text{ has failed,} \end{cases}$$

and suppose the system's structure function Ψ is defined by

$$\Psi(x_1, x_2, \dots, x_n) = \begin{cases} 1, & \text{if the system operates,} \\ 0, & \text{if the system has failed.} \end{cases} \tag{1}$$

The notion of a system signature was introduced by Samaniego (1985) for coherent systems composed of n binary components. Under the assumption of coherence and i.i.d. component lifetimes, the *system signature* $\mathbf{s}$ is an n-vector whose ith element s_i $(i = 1, 2, \dots, n)$ is the probability that the ith component failure causes the system to fail. The signature vector $\mathbf{s}$ does not depend on the common lifetime distribution of the components and is, therefore, a measure of the system design (Samaniego, 1985; Navarro et al., 2008, 2011).

The system signature is closely related to the *destruction spectrum* (Gertsbakh and Shpungin, 2009, 2011; Gertsbakh et al., 2018) and these two signatures have been used extensively in applications such as comparison of coherent systems (Navarro et al., 2008; Samaniego and Navarro, 2016; Kochar et al., 1999), lifetime estimation (Shpungin, 2007), analysis of queueing systems (Andronov et al., 2011), reliability comparison of new and used systems (Samaniego et al., 2009), analysis of failure development in connected networks (Gertsbakh and Shpungin, 2012), and evaluation of systems under minimal repair (Lindqvist and Samaniego, 2015).

2.2.1 Survival Signature

The survival signature, introduced by Coolen and Coolen-Maturi (2012), extends the system signature to the case of independent but not identically distributed component lifetimes while still isolating the system structure's contribution to reliability (Samaniego and Navarro, 2016).

We first define the survival signature for the i.i.d. case. Let the *survival signature* $\phi(\ell)$, for $\ell = 0, \ldots, n$, denote the probability that the system functions if exactly ℓ of its components function, and define Φ as the $(n+1)$-vector whose entries are $\phi(\ell)$, $\ell = 0, \ldots, n$. Let S_ℓ denote the set of state vectors in which $x_i = 1$ for exactly ℓ components, and observe that $|S_\ell| = \binom{n}{\ell}$. Because the components are i.i.d., all vectors in S_ℓ are equally likely, and therefore

$$\phi(\ell) = \binom{n}{\ell}^{-1} \sum_{\mathbf{x} \in S_\ell} \Psi(\mathbf{x}), \;\; \ell = 0, 1, \ldots, n, \tag{2}$$

as shown by Coolen and Coolen-Maturi (2012). It is straightforward to show (see Coolen and Coolen-Maturi (2012)) that the survival signature and the system signature satisfy the relationship

$$\phi(\ell) = \sum_{j=n-\ell+1}^{n} s_j, \;\; \ell = 0, 1, \ldots, n. \tag{3}$$

The right-hand side of Equation (3) denotes the probability that at least $(n - \ell + 1)$ component failures are required for the system to fail, which is equal to the probability that the system functions when exactly ℓ components function, i.e., the left-hand side of Equation (3).

Although the survival signature can be applied to systems with i.i.d. components, its fundamental contribution is the generalization of the theory of signatures to system with multiple classes of components. Following Coolen and Coolen-Maturi (2012), consider a system with $K \geq 2$ classes of components, where components of the same class have i.i.d. failure times and failure times of components of different classes are independent but not identically distributed. Let n_k denote the number of components of class k, where the n_k-values satisfy $\sum_{k=1}^{K} n_k = n$. Let $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^K)$ denote the state vector, where the subvectors $\mathbf{x}^k = (x_1^k, x_2^k, \ldots, x_{n_k}^k)$ represent the states of the components of class k.

Let $\phi(\ell_1, \ell_2, \ldots, \ell_k)$ denote the probability that a system functions if exactly $\ell_k \in \{0, 1, \ldots, n_k\}$ of its class-k components function for each $k \in \{1, 2, \ldots, K\}$. For $K > 1$, let Φ denote the *generalized survival signature*, which is a K-dimensional matrix whose entries are $\phi(\ell_1, \ell_2, \ldots, \ell_k)$ for all the values of $\ell_1, \ell_2, \ldots, \ell_K$. In what follows, we provide a mathematical characterization of the generalized survival signature.

For $k \in \{1, 2, \ldots, K\}$, let $S_\ell^k \subseteq \{0,1\}^{n_k}$ denote the set of class-k state vectors $\mathbf{x}^k$ satisfying $\sum_{i=k}^{n_k} x_i^k = \ell_k$, and observe that $|S_\ell^k| = \binom{n_k}{\ell_k}$. Let $S_\ell \subseteq \{0,1\}^n$ denote the set of whole-system state vectors $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k)$ satisfying $\mathbf{x}^k \in S_\ell^k$ for all $k \in \{1, 2, \ldots, K\}$. Given that components of class k have i.i.d. failure times, all state vectors $\mathbf{x}^k \in S_\ell^k$ are equally likely and therefore

$$\phi(\ell_1, \ell_2, \ldots, \ell_K) = \left[\prod_{k=1}^{K} \binom{n_k}{\ell_k}^{-1}\right] \times \sum_{\mathbf{x} \in S_\ell} \Psi(\mathbf{x}). \tag{4}$$

Given the survival signature Φ, Coolen and Coolen-Maturi (2012) showed that

$$P\{T > \tau\} = \sum_{\ell_1=0}^{n_1} \cdots \sum_{\ell_K=0}^{n_K} \left[\phi(\ell_1, \ldots, \ell_K) \prod_{k=1}^{K} \left(\binom{n_k}{\ell_k} [F_k(\tau)]^{n_k - \ell_k} [1 - F_k(\tau)]^{\ell_k}\right)\right], \tag{5}$$

where $F_k(\tau)$ denotes the time-to-failure CDF for components of class $k \in \{1, 2, \ldots, K\}$. Computing Equation (5) is challenging because it requires evaluating all $\prod_{k=1}^{K}(n_k + 1)$ elements of Φ.

Some notable developments of the theory of survival signature are summarized next. In Coolen et al. (2014), the authors used the survival signature in nonparametric predictive inference for system reliability. Aslett et al. (2015) applied the survival signature with Bayesian inference for system reliability quantification. Najem and Coolen (2018) employed the survival signature to study system reliability when failed components can be replaced by functioning components of the same class already in the system. Eryilmaz et al. (2018) presented general results for coherent systems with multiple classes of dependent components, and Coolen-Maturi et al. (2021) introduced a joint survival signature for multiple systems with multiple classes of components and with shared components between systems.

Eryilmaz and Tuncel (2016) generalized the survival signature to unrepairable homogeneous multi-state systems with multi-state components, and Yi et al. (2022) considered a variety of types of multi-state module systems, such as series, parallel, or recurrent structures, and derived their multi-state survival signatures in terms of the survival signatures of its modules.

Nonetheless, computing the survival signature for complex systems poses a major challenge. With this motivation, recent studies have focused on developing methods to compute or approximate the survival signature to evaluate the reliability of complex systems. Reed (2017) propose an exact algorithm combining dynamic programming and BDD to compute the survival signature of systems with multiple classes of components. This method was extended by Reed et al. (2019) for the purpose of computing the K-terminal survival signature of undirected networks with unreliable edges. Although efficient for small- and medium-sized systems, these algorithms may not be suitable for large-scale systems due to large memory requirements associated with the BDD system representation.

Xu et al. (2019) introduce an alternative method to compute the survival signature based on reliability block diagram and universal generating function. The method, however, is limited due to large memory requirements and was applied only to small networks with at most 21 components.

The literature on estimation methods for the survival signature is also in its infancy, but has received considerable attention in recent years. Behrensdorf et al. (2021) propose an approximation method that combines percolation theory and MC simulation. The authors applied this method to realistic networks such as the Berlin metro system network, which consists of 306 nodes divided into two classes and 350 edges. Already for this relatively small and sparse network, the method shows some difficulty as its algorithm takes over 27 hours to estimate the survival signature. More recently, Di Maio et al. (2023) propose a survival signature estimation method based on a combination of percolation theory with entropy-driven MC simulation. The efficiency of the method over a crude MC methods is attested through computational experiments with small networks, but the authors point out that the method may not be suited for larger networks. Lastly, whereas estimating the survival signature is the focus of the works described above, MC simulation has also been applied to estimate system reliability for a given survival signature (Patelli et al., 2017).

2.3 Other Closely Related Works

Our work builds upon prior works that have utilized an optimization subroutine in the context of evaluating network reliability by MC simulation. The work of Elperin et al. (1991) shows that MC replications for evaluating K-terminal reliability can be completed by solving a maximum-capacity spanning tree problem (using, e.g., Kruskal's algorithm). More recently, Boardman and Sullivan (2021) utilize MC simulation to evaluate the system signature with respect to reliability of guaranteeing that a minimum number of nodes remain connected to a designated sink node, and they show that MC replications can be performed by solving a one-to-all maximum-capacity path problem (using Dijkstra's algorithm). As a complementary note, it is worth noting that other versions of shortest path problem/maximum capacity path problem have also been applied in network reliability (see e.g., Ferone et al. (2021)). To our knowledge, no prior works have used optimization subroutines for the purpose of evaluating a system's survival signature.

3 Estimating the Two-terminal Survival Signature

Consider a directed network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with node set $\mathcal{N}$, where $|\mathcal{N}| = n$, and arc set $\mathcal{A}$, where $|\mathcal{A}| = m$, and let $\Gamma_i^- = \{j \in \mathcal{N} : (j,i) \in \mathcal{A}\}$ and $\Gamma_i^+ = \{j \in \mathcal{N} : (i,j) \in \mathcal{A}\}$ denote, respectively, the set of predecessors and successors of node $i \in \mathcal{N}$. We assume binary-state nodes, which fail according to a specified probability distribution. We assume arcs are perfectly reliable; however, the case of unreliable arcs and/or undirected edges can be accommodated by standard network transformations. We further assume the terminal nodes s and t cannot fail and their connectivity determines the state of the network, that is, the network is operational whenever there is a functional path from s to t and the network is failed when all the s-t paths have failed.

The non-terminal nodes $\mathcal{N} \setminus \{s,t\}$ consist of two classes of nodes where nodes within each class share a common time-to-failure distribution. Let $\mathcal{N}_e$ denote the subset of nodes in class $e \in \{1,2\}$, i.e., such that $\mathcal{N} \setminus \{s,t\} = \mathcal{N}_1 \cup \mathcal{N}_2$. We define $n_e = |\mathcal{N}_e|$ as the number of nodes in class $e \in \{1,2\}$ and we assume $n_1 \leq n_2$ without loss of generality. Let $F_e(t)$ be the common time-to-failure CDF of nodes in class e, and assume each node's time to failure is independent of any other nodes' time to failure.

For convenience, we recall the definition of the generalized survival signature for the case in hand. Here, the value $\phi(\ell_1, \ell_2)$ represents the probability that s is connected to t by a path of functioning nodes given that exactly ℓ_1 nodes from $\mathcal{N}_1$ and ℓ_2 nodes from $\mathcal{N}_2$ are functioning. The survival signature Φ is the $(n_1+1) \times (n_2+1)$ matrix whose entries are $\phi(\ell_1, \ell_2)$ for $\ell_1 = 0, 1, \ldots, n_1$, and $\ell_2 = 0, 1, \ldots, n_2$. We assume both that $\phi(0,0) = 0$, that is, the network is failed when all nodes are failed, and that $\phi(n_1, n_2) = 1$, that is, the network is functioning when all nodes are functioning. Additionally, we observe that this system cannot be deteriorated by changing a node from failed to functioning; thus, the network is a coherent system.

Given Φ, the two-terminal reliability at time $\tau > 0$ can be obtained through Equation (5) as

$$P\{T > \tau\} = \sum_{\ell_1=0}^{n_1} \sum_{\ell_2=0}^{n_2} \left[\phi(\ell_1, \ell_2) \prod_{k=1}^{2} \left(\binom{n_k}{\ell_k} [F_k(\tau)]^{n_k - \ell_k} [1 - F_k(\tau)]^{\ell_k} \right) \right]. \tag{6}$$

The main difficulty in this approach lies in obtaining the values of $\phi(\ell_1, \ell_2)$ for every combination of ℓ_1 and ℓ_2. We analyze four methods to estimate $\phi(\ell_1, \ell_2),\ \ell_1 = 0, 1, \ldots, n_1,\ \ell_2 = 0, 1, \ldots, n_2$, by using MC simulation. The methods, summarized in following subsections, differ only in how computations are performed within a MC replication. First, we summarize the common MC simulation framework of the methods.

Since nodes within the same class ($\mathcal{N}_1$ or $\mathcal{N}_2$) are equally likely to fail in any order, each of the $n_1!$ permutations of nodes in $\mathcal{N}_1$ are equally likely outcomes of the order of node failures, and similarly, there are $n_2!$ equally likely outcomes of the order of node failure in $\mathcal{N}_2$.

Thus, in each replication we independently generate a random permutation of failure times for all nodes in $\mathcal{N}_1$ and all nodes in $\mathcal{N}_2$. From each pair of simulated permutations, we extract a state vector corresponding to each pair (ℓ_1, ℓ_2) with $\ell_1 = 0, 1, \ldots, n_1$ and $\ell_2 = 0, 1, \ldots, n_2$, and assess the structure function for every one of the state vectors formed. For every pair (ℓ_1, ℓ_2), we count the number of state vectors for which the network is UP and divide this number by the number of replications generated, which gives the estimate of $\phi(\ell_1, \ell_2)$. For a given replication, let

$$\begin{aligned} q_i^1 &= k \quad \text{if } i \in \mathcal{N}_1 \text{ is the } k\text{th node to fail in } \mathcal{N}_1, \text{ and} \\ q_i^2 &= k \quad \text{if } i \in \mathcal{N}_2 \text{ is the } k\text{th node to fail in } \mathcal{N}_2. \end{aligned} \tag{7}$$

For $\ell_1 = 0, 1, \ldots, n_1$ and $\ell_2 = 0, 1, \ldots, n_2$, define $\mathbf{x}(\ell_1, \ell_2)$ such that components $i \in \mathcal{N}_1$ are UP if $q_i^1 > n_1 - \ell_1$ and DOWN otherwise, and similarly components $i \in \mathcal{N}_2$ are UP if $q_i^2 > n_2 - \ell_2$ and DOWN otherwise. Thus, the state vector $\mathbf{x}(\ell_1, \ell_2)$ represents the case where the last ℓ_1 components in the sampled permutation of $\mathcal{N}_1$ and the last ℓ_2 components in the sampled permutation of $\mathcal{N}_2$ are UP, and all remaining components are DOWN.

Following the above procedure, each MC replication $j = 1, 2, \ldots, \mathsf{M}$ yields a state vector $\mathbf{x}_j(\ell_1, \ell_2)$ for every pair (ℓ_1, ℓ_2). In replication j, we evaluate $\Psi(\mathbf{x}_j(\ell_1, \ell_2))$ for all $\ell_1 = 0, 1, \ldots, n_1$ and $\ell_2 = 0, 1, \ldots, n_2$, where $\Psi(\mathbf{x}_j(\ell_1, \ell_2)) = 1$ if there exists a path of functioning nodes from s to t in state $\mathbf{x}_j(\ell_1, \ell_2)$, and $\Psi(\mathbf{x}_j(\ell_1, \ell_2)) = 0$ otherwise. Hereafter, we refer to the collection $\Psi(\mathbf{x}_j(\ell_1, \ell_2))$, $\ell_1 = 0, 1, \ldots, n_1$; $\ell_2 = 0, 1, \ldots, n_2$, as the *system state matrix* for replication j. After completing M MC replications, we estimate $\phi(\ell_1, \ell_2)$ by

$$\phi(\ell_1, \ell_2) = \frac{\sum_{j=1}^{\mathsf{M}} \Psi(\mathbf{x}_j(\ell_1, \ell_2))}{\mathsf{M}}. \tag{8}$$

In what follows, we first present an intuitive approach (hereafter referred to as the *Naive* approach), based on breadth-first search (BFS), for evaluating the system state matrix in each replication. Then, we discuss the *incremental search* approach, which improves upon the naive algorithm by leveraging the nondecreasing nature of the structure function of coherent systems in order to avoid performing redundant searches. The third method is an extension of the method of Boardman and Sullivan (2021), and hence we call it the *single-objective optimization* approach. The last method, called the *bi-objective optimization* approach, is based on solving a bi-objective maximum capacity path problem at every replication and is our main contribution.

3.1 Naive Approach

In this approach, the value of $\Psi(\mathbf{x}_j(\ell_1, \ell_2))$ is evaluated for each $\ell_1 = 0, 1, \ldots, n_1$ and $\ell_2 = 0, 1, \ldots, n_2$ by performing BFS (see, e.g., Ahuja et al. (1993)); that is, BFS is run a total of $\mathsf{M} \times (n_1 + 1) \times (n_2 + 1)$ times. Because each BFS has $O(m)$ time complexity, the resulting complexity of the MC algorithm is $O(n_1 n_2 m \mathsf{M})$. The Naive approach is summarized in Algorithm 5 in Appendix A.

3.2 Incremental Search Approach

The incremental_search (IS) method exploits the fact that $\mathbf{x}_j(\ell_1, \ell_2)$ and $\mathbf{x}_j(\ell_1, \ell_2 + 1)$ are identical except for the addition of the node $q^2_{n_2-\ell_2}$. Using this property, IS computes an entire row of the system state matrix (i.e., $\Psi(\mathbf{x}_j(\ell_1, -))$ for a given value ℓ_1) in the same worst-case time required to evaluate a single element of the system state matrix in Naive. We explain this approach in the following paragraph.

For fixed ℓ_1, steps 8–16 of Algorithm 5 compute the row $\Psi(\mathbf{x}_j(\ell_1, -))$ by running BFS $n_2 + 1$ times. The IS method is identical to Algorithm 5 with the exception that we do not re-initialize BFS for (ℓ_1, ℓ_2) with $\ell_2 > 1$; rather, we update the search from $(\ell_1, \ell_2 - 1)$ to include the new node that was turned UP in $\mathcal{N}_2$. The update can be done by checking whether the new node contains an incoming arc from any of the nodes already marked in the search. If so, we add the new UP node to the list of nodes to explore and continue the search as if the referred node had been encountered in the original search. For fixed ℓ_1, this modified search algorithm encounters each arc at most twice. To see this, observe that the predecessor list Γ^-_i of each node i is scanned at most once (i.e., after the node i is turned UP) and the successor list Γ^+_i is also scanned at most once (i.e., after the node i is marked). For fixed ℓ_1, a total of $O(m)$ effort is required because each arc is examined at most twice. The total work required in each replication is therefore $O(n_1 m)$ and the worst-case complexity of IS is $O(n_1 m \mathsf{M})$.

3.3 Single-objective Optimization Approach

Although IS is more efficient than Naive in that the first avoid performing many of the redundant steps performed by the latter, both approaches follow the same principle. They both compute a row of the system state matrix by adding nodes, according to the permutation of $\mathcal{N}_2$, until the network is connected. The single-objective optimization approach (SO) is based on a different idea; that of solving an optimization problem in order to directly determine how many nodes must be removed, according to the permutation of $\mathcal{N}_2$, to disconnect the network assuming that all nodes in $\mathcal{N}_2$ are operational at the beginning.

In SO, we extend the work of Boardman and Sullivan (2021) by solving an optimization problem for each value of ℓ_1 to find the corresponding value of ℓ_2 for which the network fails. We now explain the main ideas.

Consider Algorithm 5 again, and suppose we fix the value of ℓ_1 in the jth replication. In place of steps

7–13, we need only to identify the maximum value of ℓ_2, ${\ell_2}^*$, for which $\Psi(\mathbf{x}_j(\ell_1,\ell_2)) = 0$. Once ${\ell_2}^*$ is obtained, we have

$$\begin{aligned}&\Psi(\mathbf{x}_j(\ell_1,0)) = \Psi(\mathbf{x}_j(\ell_1,1)) = \cdots = \Psi(\mathbf{x}_j(\ell_1,{\ell_2}^*)) = 0, \text{ and}\\ &\Psi(\mathbf{x}_j(\ell_1,{\ell_2}^*+1)) = \Psi(\mathbf{x}_j(\ell_1,\ell_2+2)) = \cdots = \Psi(\mathbf{x}_j(\ell_1,n_2)) = 1,\end{aligned} \tag{9}$$

since Ψ is nondecreasing in $\mathbf{x}$.

The problem of finding ${\ell_2}^*$ can be formulated as an instance of the single objective maximum capacity path problem (Pollack, 1960; Hu, 1961). This idea was shown initially by Boardman and Sullivan (2021) in the context of a similar problem involving i.i.d. components. For fixed ℓ_1, we can adapt their approach to find ${\ell_2}^*$. The single-objective optimization approach is based on solving the single-objective maximum capacity path problem once for each value of $\ell_1 = 0, 1, \ldots, n_1$, with the additional consideration that the ℓ_1 components from $\mathcal{N}_1$ that are UP are uncapacitated, which we represent by assigning "∞" as their capacity.

To formalize the single-objective optimization approach, independently simulate a permutation of $\mathcal{N}_1$ and a permutation of $\mathcal{N}_2$, and record these permutations according to Equation (7). Then, for fixed ℓ_1, associate a weight u_i to each node $i \in \mathcal{N}$ according to

$$u_i = \begin{cases} q_i^2, & i \in \mathcal{N}_2,\\ 0, & \text{if } i \in \mathcal{N}_1 \text{ and } q_i^1 \le n_1 - \ell_1,\\ \infty, & \text{otherwise.}\end{cases} \tag{10}$$

In practice, we substitute ∞ by a number larger than or equal to $\max\{q_i^2 : i \in \mathcal{N}_2\}$, such as the total number of nodes, n, in the network. Let $\mathcal{P}$ denote the set of all directed paths p from s to t. The value ${\ell_2}^*$ is then obtained by solving the maximum capacity path (MCP) problem

$$v^* = \max_{p\in\mathcal{P}}\{\min\{u_k : k \in p\}\}, \tag{11}$$

and setting ${\ell_2}^* = n_2 - v^*$. In Equation (11), $\min\{u_k : k \in p\}$ represents the number of nodes (within the simulated permutation of $\mathcal{N}_2$) that must fail to disconnect path p, and v^* thus represents the number of node failures needed to cause system failure.

Pollack (1960) observed that the MCP problem can be solved using a slight modification of Dijkstra's algorithm. In the case of the node-capacitated MCP problem of Equation (11), Dijkstra's algorithm can be applied by initializing node labels as $\mathsf{d}(i) = 0$ for all $i \in \mathcal{N} \setminus \{s\}$, and $\mathsf{d}(s) = \infty$, and then updating (when considering an arc (i, j) leaving a node i whose label has been made permanent) according to

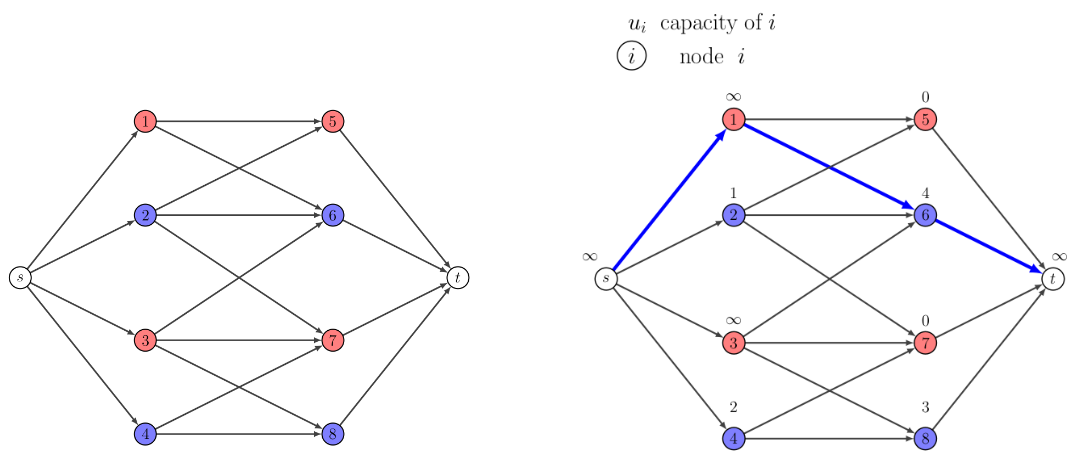
$$\mathsf{d}(j) = \max\{\mathsf{d}(j), \min\{\mathsf{d}(i), u_j\}\}. \tag{12}$$

Additionally, a node label with maximum value is made permanent in each iteration instead of minimum value, as in the case of the shortest path.

Algorithm 6 presents the pseudocode for our implementation of Dijkstra's algorithm, which is an adaptation of the Dials-Dijkstra algorithm presented in Ahuja et al. (1993). Dial's implementation of Dijkstra's algorithm is motivated for our problem because each label is bounded between 0 and n; thus, temporarily labeled nodes can be stored in a sorted fashion using a small number of "bucket" sets, which allows the algorithm to avoid scanning all temporarily labeled nodes at each iteration. The algorithm keeps $n + 1$ bucket sets numbered 0 through n, where bucket k stores all nodes with temporary label equal to k. The while loop (lines 6–21) of Algorithm 6 identifies the greatest-numbered nonempty bucket. Once this bucket is found, any node in it has its label made permanent and its outgoing arcs explored to propose new temporary labels for adjacent nodes, possibly resulting in moving the nodes to a bucket set of increased number (lines 8–18).

Algorithm 7 states the SO approach. Noting that the Dials-MaxCapPath algorithm provides ${\ell_2}^*$ for each value of ℓ_1, steps 8–12 of Algorithm 7 populate the corresponding row $\Psi(\mathbf{x}_j(\ell_1, -))$ according to Equation (9) by simply adding 1 to any entry for which $\ell_2 > {\ell_2}^*$. We demonstrate SO with the following example.

Example 1. Consider the network in Figure 1(a). For this network, $n = 10$ nodes, $\mathcal{N}_1 = \{1, 3, 5, 7\}$ is represented in red, and $\mathcal{N}_2 = \{2, 4, 6, 8\}$ is represented in blue. Suppose that in the jth replication of Algorithm 7, we generate the permutation $P_1 = \{5, 7, 3, 1\}$ for $\mathcal{N}_1$, and $P_2 = \{2, 4, 8, 6\}$ for $\mathcal{N}_2$. Therefore, for these permutations, node 5 is the first node to fail in $\mathcal{N}_1$, followed by nodes 7, 3, and 1, respectively, and node 2 is the first node to fail in $\mathcal{N}_2$, followed by nodes 4, 8, and 6, respectively.



(a) Heterogeneous s-t network.

(b) Network for $\ell_1 = 2$.

Figure 1: Computation of system state matrix row corresponding to $\ell_1 = 2$ using SO.

For every value of ℓ_1, the algorithm associates to each node $i \in \mathcal{N}$ a capacity u_i according to Equation (10); for $\ell_1 = 2$, the algorithm obtains the network in Figure 1(b). Then, the algorithm solves the MCP problem. At termination, Algorithm 6 provides a tree of maximum capacity paths rooted in s, and $v^* = 4$ is the capacity of the s-t path in this tree (highlighted in blue in Figure 1(b)). Therefore, for any $\ell_2 > {\ell_2}^*$ $(= n_2 - v^* = 4 - 4 = 0)$, the network is UP. The resulting system state matrix is shown in Table 1, where the row corresponding to $\ell_1 = 2$ is highlighted in blue. Observe that the only value of ℓ_2 for which the network is DOWN is 0, which corresponds to ${\ell_2}^* = 0$, and is formatted in red. □

Table 1: System state matrix for permutations $P_1 = \{5, 7, 3, 1\}$ and $P_2 = \{2, 4, 8, 6\}$ computed with SO.

	$\Psi(\mathbf{x}_j(\ell_1, \ell_2))$				
$l_1 \backslash l_2$	0	1	2	3	4
0	0	0	0	1	1
1	0	1	1	1	1
2	**0**	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1

Proposition 3.1 states the complexity of the SO approach and its proof is presented in Appendix C. Notice that the complexity of the SO approach is equal to the complexity of the IS approach.

Proposition 3.1. *The overall complexity of the* SO *approach is* $O(n_1 n\mathsf{M} + n_1 m\mathsf{M}) = O(n_1 m\mathsf{M})$ *provided that* $n < m$.

3.4 Bi-objective Optimization Approach

3.4.1 Modeling of the Bi-objective Optimization Approach

Whereas SO solves an optimization problem for every row of the system state matrix, the bi-objective optimization approach (BO) extends the idea of SO by solving a single bi-objective optimization problem to evaluate the entire system state matrix in each replication.

As with the previous approaches, we begin each replication by generating $q_i^1, i \in \mathcal{N}_1$, and $q_i^2, i \in \mathcal{N}_2$ according to Equation (7). To every node $i \in \mathcal{N}$, associate two weights according to

$$u_i^e = \begin{cases} q_i^e, & i \in \mathcal{N}_e, \\ \infty, & i \in \mathcal{N}/\mathcal{N}_e, \end{cases} \qquad e = 1, 2. \tag{13}$$

Again, in our implementation, we substitute ∞ by n, and hence the largest value of u_i^e, $e = 1, 2$, is n.

To compute the system state matrix for the jth replication, we must determine all pairs (ℓ_1, ℓ_2) for which the network is DOWN. In BO, we determine these combinations by solving the bi-objective maximum capacity path problem (BOMCP), which can be defined as follows. For a network with weights u_i^1 and u_i^2 associated to each node $i \in \mathcal{N}$, let $\mathcal{P}$ denote the set of all s-t paths. For $p \in \mathcal{P}$, define capacities

$$c^1(p) := \min\{u_i^1 : i \in p\} \quad \text{and} \quad c^2(p) := \min\{u_i^2 : i \in p\}.$$

Similarly to SO, these capacities represent the number of failures in $\mathcal{N}_1$ and in $\mathcal{N}_2$ that disconnects path p.

Define a path p from s to i as a non-dominated s-i path if there does not exist any other path p' from s to i such that $c^1(p') \geq c^1(p)$ and $c^2(p') \geq c^2(p)$ with at least one strict inequality, and define a non-dominated s-i point as the image of a non-dominated s-i path p under c^1 and c^2. Then, the BOMCP problem can be defined as

$$\max_{p \in \mathcal{P}} \{c^1(p), c^2(p)\}, \tag{14}$$

and a solution to this problem provides a set Ω of non-dominated s-t points (v_1^*, v_2^*), which can be used to derive the values of (ℓ_1, ℓ_2) for which $\Psi(\mathbf{x}(\ell_1, \ell_2)) = 0$, for all $\ell_1 = 0, 1, \ldots, n_1$ and $\ell_2 = 0, 1, \ldots, n_2$. Letting

(v_1^*, v_2^*) denote such a non-dominated point, the network is UP (i.e., $\Psi(\mathbf{x}(\ell_1, \ell_2)) = 1$) for every point (ℓ_1, ℓ_2) with $v_1^* > n_1 - \ell_1$ and $v_2^* > n_2 - \ell_2$. Furthermore, the existence of such a non-dominated point is guaranteed for any (ℓ_1, ℓ_2) in which $\Psi(\mathbf{x}(\ell_1, \ell_2)) = 1$. We record this result in the following theorem.

Theorem 3.1. *$\Psi(\mathbf{x}(\ell_1, \ell_2)) = 1$ if and only if there exists a non-dominated point (v_1^*, v_2^*) such that $v_1^* > n_1 - \ell_1$ and $v_2^* > n_2 - \ell_2$.*

Proof. ($\Rightarrow$) Suppose $\Psi(\mathbf{x}(\ell_1, \ell_2)) = 1$. Then by definition of $\mathbf{x}(\ell_1, \ell_2)$, the system is UP when all components $i \in \mathcal{N}_1$ with $q_i^1 > n_1 - \ell_1$, and all components $i \in \mathcal{N}_2$ with $q_i^2 > n_2 - \ell_2$ are UP, and the remaining components in $\mathcal{N}_1$ and $\mathcal{N}_2$ are DOWN. Thus, there exists an s-t path p such that $c^1(p) > n_1 - \ell_1$ and $c^2(p) > n_2 - \ell_2$. Either p is a non-dominated path or it is dominated by some other s-t path, and there exists a non-dominated point (v_1^*, v_2^*) with $v_1^* \geq c^1(p)$ and $v_2^* \geq c^2(p)$. Therefore, $v_1^* > n_1 - \ell_1$ and $v_2^* > n_2 - \ell_2$.
($\Leftarrow$) Conversely, suppose that there exists a non-dominated point (v_1^*, v_2^*) such that $v_1^* > n_1 - \ell_1$ and $v_2^* > n_2 - \ell_2$. Then, there exists an s-t path p with capacity $c^1(p) = v_1^*$ and $c^2(p) = v_2^*$, and hence $c^1(p) > n_1 - \ell_1$ and $c^2(p) > n_2 - \ell_2$. Thus, for all $i \in p$, $q_i^1 > n_1 - \ell_1$ and $q_i^2 > n_2 - \ell_2$, that is, all components $i \in p$ are UP with respect to the state $\mathbf{x}(\ell_1, \ell_2)$. Because $\mathbf{x}(\ell_1, \ell_2)$ contains an s-t path of functioning components, $\Psi(\mathbf{x}(\ell_1, \ell_2)) = 1$. $\square$

3.4.2 Solution Methodology

We solve the BOMCP using the $\mathsf{BO\text{-}MaxCapPath}$ algorithm (given in Algorithm 1) by modifying the "BDijkstra" bi-objective shortest path algorithm of Sedeño-Noda and Colebrook (2019) in essentially the same way as Dijkstra's algorithm (for single-objective shortest path) is modified for MCP.

In Algorithm 1, labels associated with each node $i \in \mathcal{N}$ contain the value of both $c^1(p)$ and $c^2(p)$ for a candidate s-i path, i.e., a potential non-dominated s-i path. Let d_i^1 and d_i^2 respectively denote stored values of $c^1(p)$ and $c^2(p)$ for the candidate s-i path, where the values $\mathsf{d}_s^1 = \mathsf{d}_s^2 = \infty$ and $\mathsf{d}_i^1 = \mathsf{d}_i^2 = 0$ are initialized in similar fashion to Algorithm 6. Let Ω_i denote the set of non-dominated s-i points and, without loss of generality, assume that the network contains a path from s to every other node $i \in \mathcal{N} \setminus \{s\}$. Then, the BOMCP problem satisfies the following principle of optimality: for every node i and every non-dominated s-i point $(\mathsf{d}^1, \mathsf{d}^2) \in \Omega_i$, there exists a path from s to i with capacities $(\mathsf{d}^1, \mathsf{d}^2)$ that can be composed as a non-dominated path from s to some node $j \in \Gamma_i^-$ plus the arc (j, i). With this, after a candidate s-i path p is determined to be a non-dominated s-i path, the values $(\mathsf{d}_i^1, \mathsf{d}_i^2)$ are recorded (i.e., made permanent) as a non-dominated s-i point and a new label is proposed for each node $j \in \Gamma_i^+$ based on adding the arc (i, j) to the end of p in similar fashion to Equation (12).

The correctness of the BDijkstra algorithm has been proved for the bi-objective shortest path problem (Sedeño-Noda and Colebrook, 2019). Because our extension of this algorithm is analogous to the extension of Dijkstra's algorithm from (single-objective) shortest path to MCP, we have not proven correctness here. Besides the previously mentioned initialization of d_i^1 and d_i^2, $i \in \mathcal{N}$, the differences between Algorithm 1 and BDijkstra are as follows: (i) Algorithm 1 extracts labels from the heap based on the lexicographic maximum key $(\mathsf{d}_i^1, \mathsf{d}_i^2)$ instead of the lexicographic minimum key used by BDijkstra; (ii) Algorithm 1 proposes new candidate labels for a node j based upon the minimum c^1 and c^2 capacities, respectively, in the non-dominated s-j path instead of adding the lengths corresponding to each objective of each arc in the non-dominated s-j path, as in BDijkstra; and (iii), Algorithm 1 accepts a new candidate label for node j if it lexicographically increases (instead of lexicographically decreases as in BDijkstra) the current label and is not dominated by a permanent label for node j. We now explain the extension of BDijkstra for BOMCP.

Algorithm 1: BO-MaxCapPath

```
1  $\mathsf{N}_i \leftarrow 0$, $\mathsf{d}_i^1 \leftarrow 0$, $\mathsf{d}_i^2 \leftarrow 0$, InH$[i] \leftarrow$ False, $i \in \mathcal{N} \setminus \{s\}$
2  $\mathsf{N}_s \leftarrow 0$, $\mathsf{d}_s^1 \leftarrow \infty$, $\mathsf{d}_s^2 \leftarrow \infty$, $l \leftarrow (s, \infty, \infty, -, -)$
3  insert($l$, H); InH$[s] \leftarrow$ True
4  while H $\neq \emptyset$ do
5      $l^* \leftarrow$ find-max(H), delete-max(H)
6      $\mathsf{d}_{i^*}^1 \leftarrow 0$, $\mathsf{d}_{i^*}^2 \leftarrow 0$                // $i^*$ is the node with label $l^*$
7      $\mathsf{N}_{i^*} \leftarrow \mathsf{N}_{i^*} + 1$, L$[i^*][\mathsf{N}_{i^*}] \leftarrow l^*$, InH$[i^*] \leftarrow$ False
8      $l^{new} \leftarrow$ NewCandidateLabel($i^*$, $l^*$)
9      if $l^{new} \neq$ Null then
10         insert($l^{new}$, H), InH$[i^*] \leftarrow$ True
11         $\mathsf{d}_{i^*}^1 \leftarrow l^{new}.\mathsf{d}^1$, $\mathsf{d}_{i^*}^2 \leftarrow l^{new}.\mathsf{d}^2$
12     end
13     RelaxationProcess($i^*$, H, $l^*$)
14 end
15 return L$[t]$
```

Following the notation of Sedeño-Noda and Colebrook (2019), the data structure used in Algorithm 1 to

Algorithm 2: NewCandidateLabel(i^*, l^*)

```
d$^1$ ← 0, d$^2$ ← 0; $l^{new}$ ← Null          // $\Gamma_i^-$ is the set of predecessors of node $i$
for $j \in \Gamma_{i^*}^-$ do
    for $l \in$ L[$j$] do
        $f^1 \leftarrow \min\{l.d^1, u_{i^*}^1, u_j^1\}$
        $f^2 \leftarrow \min\{l.d^2, u_{i^*}^2, u_j^2\}$
        if $f^1 > d^1$ or $f^1 = d^1$ and $f^2 > d^2$          // lexmax cand label
         then
            if $f^1 < l^*.d^1$ and $f^2 > l^*.d^2$          // non-dom cand label
             then
                $d^1 \leftarrow f^1$; $d^2 \leftarrow f^2$
                $l^{new} \leftarrow (i^*, d^1, d^2, j, r)$          // $r$ is the position of $l$ in L[$j$]
            end
        end
    end
end
return $l^{new}$
```

Algorithm 3: RelaxationProcess(i^*, H, l^*)

```
for $j \in \Gamma_{i^*}^+$ do
    $f^1 \leftarrow \min\{l^*.d^1, u_{i^*}^1, u_j^1\}$
    $f^2 \leftarrow \min\{l^*.d^2, u_{i^*}^2, u_j^2\}$
    if $f^1 > d_j^1$ or $f^1 = d_j^1$ and $f^2 > d_j^2$          // Relaxation ($i^*$, $j$)
     then
        if $N_j = 0$ or $f^1 < L[j][N_j].d^1$ and $f^2 > L[j][N_j].d^2$          // non-dom. label
         then
            $d_j^1 \leftarrow f^1$; $d_j^2 \leftarrow f^2$
            $l \leftarrow (j, d_j^1, d_j^2, i^*, N_{i^*})$          // $N_{i^*}$ is the position of $l^*$ in L[$i^*$]
            if InH[$j$] = False then
                insert($l$, H)
                InH[$j$] ← True
            end
            else
                increase-key($l$, H)
            end
        end
    end
end
```

store the non-dominated points for $i \in \mathcal{N}$ is denoted by $\mathsf{L}[i]$. Since multiple non-dominated points may be associated with each node $i \in \mathcal{N}$, $\mathsf{L}[i]$ is dynamically increased by one point each time a new non-dominated point associated with i is found. The total number of non-dominated points associated with i, N_i, is not known until termination. At termination, $\mathsf{L}[i]$ contains non-dominated points $\mathsf{L}[i][1]$, $\mathsf{L}[i][2], \dots, \mathsf{L}[i][\mathsf{N}_i]$, stored in lexicographically decreasing order. Non-dominated points are represented by labels of the form

$$(i, \mathsf{d}^1, \mathsf{d}^2, j, r), \tag{15}$$

where i denotes the node to which the non-dominated point is associated, d^1 and d^2 denote, respectively the capacity of node i for the first and second objectives, j denotes the predecessor of node i in the respective non-dominated path, and r denotes the position in $\mathsf{L}[i]$ of the non-dominated label j that allows the corresponding non-dominated path to i to be obtained.

As in the BDijkstra algorithm of Sedeño-Noda and Colebrook (2019), Algorithm 1 maintains a heap, H, that stores at most one candidate label for each node $i \in \mathcal{N}$, and therefore has a maximum size of n. Every label has an associated key given by the pair $(\mathsf{d}^1, \mathsf{d}^2)$. The candidate label $l \in \mathsf{H}$ associated with node i is not in $\mathsf{L}[i]$ since a label is stored in L only when the label becomes permanent. Similarly to BDijkstra, our algorithm maintains the invariant that the key of a label l in H associated with node i is not dominated by the key of any label in $\mathsf{L}[i]$, for all $i \in \mathcal{N}$. Additionally, the key of the candidate label for node i is the lexicographic maximum among all paths to node i that can be created by a known non-dominated path to some predecessor node $j \in \Gamma_i^-$ plus the arc (j, i). The heap performs the following basic operations on labels: $\mathsf{find\text{-}max(H)}$, $\mathsf{delete\text{-}max(H)}$, $\mathsf{insert}(l,\ \mathsf{H})$, and $\mathsf{increase\text{-}key}(l,\ \mathsf{H})$, and labels are extracted from the heap in lexicographic maximum order of their keys, i.e., a label $l^* = (i, \mathsf{d}^1, \mathsf{d}^2, j, r)$ is extracted from the heap if, for any other label l in the heap, $l^*.\mathsf{d}^1 > l.\mathsf{d}^1$ or $l^*.\mathsf{d}^1 = l.\mathsf{d}^1$ and $l^*.\mathsf{d}^2 > l.\mathsf{d}^2$.

For a label l^* associated with a node i^* to become permanent (i.e., recording its key as a non-dominated s-i^* point) it has to satisfy two conditions: (1) its key must not be dominated by the key of any label already in $\mathsf{L}[i^*]$; and (2) there does not exist a non-explored path from s to i^* whose key dominates the key of l^*. Item (1) is satisfied by any label in H at any time by the invariant already discussed. Due to the label update procedures NewCandidateLabel (Algorithm 2) and RelaxationProcess (Algorithm 3), it can be shown that item (2) is satisfied by a label l^* extracted from the heap. Therefore, the label l^* extracted from the heap in an iteration becomes permanent and it is added to the end of $\mathsf{L}[i^*]$. In this way, the key of a new permanent label associated with a node i^* is non-dominated by and lexicographically smaller than any other permanent label in $\mathsf{L}[i^*]$.

With the exception of the first label, $(s, \infty, \infty, -, -)$, any other label is generated by either NewCandidateLabel or RelaxationProcess from a permanent label. When a label l^* associated with a node $i^* \in \mathcal{N}$ is made permanent, NewCandidateLabel determines whether there is another explored path to node i^* that is not dominated by any point already in $\mathsf{L}[i^*]$, and RelaxationProcess explores all s-j paths by extending the non-dominated s-i^* path corresponding to l^* by a single arc (i^*, j), where $j \in \Gamma^+_{i^*}$.

If NewCandidateLabel finds a new s-i^* path and/or RelaxationProcess finds a new non-dominated s-j path, $j \in \Gamma^+_{i^*}$, a new label is created and inserted into the heap. Furthermore, since the labels are made permanent in decreasing order and a permanent label associated with a node i^* is added to the end of $\mathsf{L}[i^*]$, in the dominance test for a new label l^* it is only necessary to check whether the key of the last label in $\mathsf{L}[i^*]$ dominates the key of l^*.

Solving the BOMCP provides the set of non-dominated points Ω. Then, we can update Φ by looping over Ω and adding 1 to every entry that satisfies the condition $v_1^* > n_1 - \ell_1$ and $v_2^* > n_2 - \ell_2$. Algorithm 4 states the BO approach.

We establish the complexity of the bi-objective optimization approach with Proposition 3.2, and Corollary 3.1.1, and we refer the reader to Appendix D for the proof of 3.2. We illustrate the BO approach with the example below.

Proposition 3.2. *The complexity of Algorithm 1 is* $O(n_1^2 m + n_1 m \log n)$.

Corollary 3.1.1 follows directly from the fact that Algorithm 4 solves the BOMCP problem for each replication.

Corollary 3.1.1. *The complexity of the bi-objective optimization approach is* $O(n_1^2 m\mathsf{M} + n_1 m \log n\mathsf{M})$.

Example 1 (continued)**.** Consider the network in Figure 1(a). For this network, $n = 10$, $\mathcal{N}_1 = \{1, 3, 5, 7\}$ and $\mathcal{N}_2 = \{2, 4, 6, 8\}$. Suppose that in the jth replication, the algorithm generates the permutations $P_1 =$

Algorithm 4: BO

```
$\Phi \leftarrow \mathbf{0}$
for $j = 1$ to M do
    $\triangleright$ Simulate a permutation of $\mathcal{N}_1$ and a permutation of $\mathcal{N}_2$
    $\triangleright$ Update $u_i,\ i \in \mathcal{N}$, according to Equation (13)
    $\Omega \leftarrow$ BO-MaxCapPath($\mathcal{G}$)                // $\Omega$ stores all $(v_1^*, v_2^*)$ points
    for $(v_1^*, v_2^*) \in \Omega$ do
        for $\ell_1 = 0$ to $n_1$ do
            for $\ell_2 = 0$ to $n_2$ do
                if $v_1^* > n_1 - \ell_1$ and $v_2^* > n_2 - \ell_2$ then
                    $\phi(\ell_1, \ell_2) \leftarrow \phi(\ell_1, \ell_2) + 1$
                end
            end
        end
    end
end
$\Phi \leftarrow \Phi/$M.
return $\Phi$
```

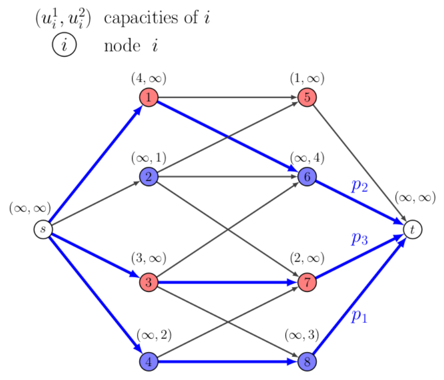


Figure 2: Network after the updating of u_i.

$\{5, 7, 3, 1\}$ and $P_2 = \{2, 4, 8, 6\}$ for $\mathcal{N}_1$ and $\mathcal{N}_2$, respectively. The algorithm then updates the values of u_i^1 and u_i^2, $i \in \mathcal{N}$, according to Equation (13) (see Figure 2), and solves the corresponding BOMCP problem. The solution of the BOMCP problem shows that there are three non-dominated s-t paths (marked in blue in Figure 2): $p_1 = s$ - 4 - 8 - t, $p_2 = s$ - 1 - 6 - t, and $p_3 = s$ - 3 - 7 - t, with respective non-dominated points $(\infty, 2)$, $(4, 4)$, and $(2, \infty)$; these are the points stored in Ω. The algorithm then loops over these points populating the survival signature matrix. Consider the first point stored in Ω, $(\infty, 2)$. The network is UP for every point (ℓ_1, ℓ_2) such that $\infty > 4 - \ell_1$ and $2 > 4 - \ell_2$, that is, the network is UP for $\ell_1 > -\infty$

($\ell_1 \geq 0$) and $\ell_2 > 2$ (this area is outlined in red in Table 2). Similarly, point $(v_1^*, v_2^*) = (4, 4)$ (yellow) and $(v_1^*, v_2^*) = (2, \infty)$ (blue). The brown region is where the three other regions overlap. $\square$

Table 2: System state matrix for permutations $P_1 = \{5, 7, 3, 1\}$ and $P_2 = \{2, 4, 8, 6\}$ computed with BO.

	$\Psi(\mathbf{x}_j(\ell_1, \ell_2))$				
$l_1 \backslash l_2$	0	1	2	3	4
0	0	0	0	1	1
1	0	1	1	1	1
2	0	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1

4 Computational Experiments

4.1 Implementation Details and Experiments Setup

We implemented the four methods in Section 3 and published the codes, as well as all the results from the experiments discussed in this section, at the `dblsBR/Heterogeneous_Signature` repository. All algorithms were implemented in C++, and all experiments were performed on an Intel® Core i7-1165G7 CPU laptop with a 2.80 GHz processor and 16 GB RAM running on Windows 10 OS.

To validate our methods, we (i) verified that the estimated survival signature produced by all four methods were equivalent for all instances; (ii) verified that the methods produced the (exact) survival signature for a small network commonly used to validate exact and estimating methods (see Patelli et al. (2017); Behrensdorf et al. (2021); Coolen-Maturi et al. (2021); Di Maio et al. (2023)) — see Figure 3 for a diagram and Table 3 for the corresponding survival signature) when the MC simulation was configured to generate each pair of permutations (one from $\mathcal{N}_1$ and one from $\mathcal{N}_2$) once; and (iii) created a non-trivial system for which we can compute the exact survival signature (by observing properties of the system's topology), and therefore assess the accuracy of our method through hypothesis tests.

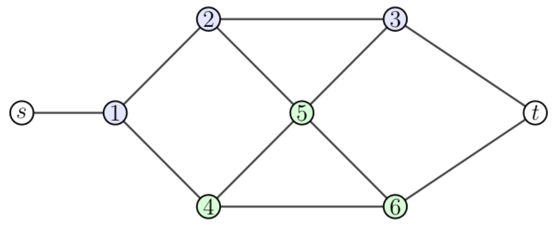


Figure 3: Bridge system from Patelli et al. (2017), where $\mathcal{N}_1 = \{1, 2, 3\}$ and $\mathcal{N}_2 = \{4, 5, 6\}$

$\ell_1 \backslash \ell_2$	0	1	2	3
0	0	0	0	0
1	0	0	1/9	1/3
2	0	0	4/9	2/3
3	1	1	1	1

Table 3: Survival signature of Figure (3).

For item (iii) of the preceding list, i.e., the accuracy of our methods, we first computed the exact survival signature of two instances of a *chain* network (see Figure 6 for an illustration of one of the instances), one small instance with 25 nodes and a larger instance with 61 nodes. Then, we estimated the signature for each instance (with 20000 replications for the small instance and 75000 replications for the larger instance) and conduct a hypothesis test on proportion where we test the hypothesis that the true MC estimated signature was equal to the exact signature. For a level of significance $\alpha = 0.05$, the rejection rates were 3.57% for the small instance and 3.16% for the larger instance, and therefore very close to the 5% rejection rate expected in this setting. The details about the computation of the exact signatures and the hypothesis tests are discussed in details in Appendix E .

We performed two sets of experiments. The first set focused on evaluating the performance of each approach for a varying number of replications as well as on verifying that the running time of each approach increases approximately linearly with the number of replications. Hence, we compare the running time of each approach for a varying M with a fixed rate of n_1/n_2. Once we verify the linearity of the running time with M, we can compute the rate of time per replication based on a relatively small number of replications (say 5000), and then estimate the total running time for a large number of replications (say 1000000). For this set we considered medium-sized systems commonly used in reliability analysis.

The second set of experiments focused on comparing the performance of the approaches for a fixed number of replications and a varying rate n_1/n_2. For this set of experiments, we considered a relatively extensive collection of instances: two medium-sized systems commonly used in reliability analysis, a large-scale, generally structured system with multiple density rates, and a large-scale realistic power system. We introduce each test instance in the next section.

4.2 Test Instances

We have employed a variety of network systems with different sizes and from different areas of application. For the first set of experiments, we adapted two medium-sized networks commonly used in reliability studies from Sebastio et al. (2014).

The first network is a communication system, has $n = 59$ nodes, $m = 142$ arcs, and is composed of three replicas of the 1973 Arpanet network (see Figure 4(a)). We adapted this network in the following fashion: the 57 nodes in $\mathcal{N} \setminus \{s, t\}$ (numbered from 1 to 57) are divided into two classes of nodes such that $n_1 = 28$ first even-numbered nodes are assigned to class 1 (i.e., $2, 4, \ldots, 56$) and the remaining $n_2 = 29$ nodes are assigned to class 2.

The second network is a typical electrical system of an airplane and is depicted in Figure 4(b). The airplane system has $n = 82$ nodes and $m = 268$ arcs. We divided the nodes in $\mathcal{N} \setminus \{s, t\}$ into $\mathcal{N}_1$ and $\mathcal{N}_2$ with $n_1 = 40$ and $n_2 = 40$ in a similar fashion to what was done for the Arpanet system.

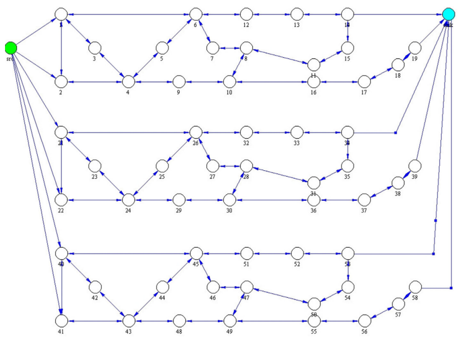
The Arpanet system and the Airplane electrical system were used both in the first and the second set of experiments. However, to provide a comprehensive evaluation of the performance of each approach, we considered additional networks in the second set of experiments. These additional networks are presented next.

We also considered more challenging test instances. We started by considering large-scale, generally structured graphs of theoretical interest. We generated a random geometric graph (RGG) with 350 nodes according to the following procedure: in the X-Y plane, we located node s with coordinates $x_s = 0$ and $y_s = 10$ and node t with coordinates $x_t = 10$ and $y_t = 0$. For any other node $i \in \mathcal{N} \setminus \{s, t\}$, we randomly generated coordinates x_i and y_i between 0 and 10 according to a uniform distribution, and we created an arc from node i to node $j \in \mathcal{N}$ if and only if the Euclidean distance between i and j is smaller than or equal to a parameter d. Notice that this procedure does not prevent the creation of cycles, which adds another layer of generality to the RGG. We then assign nodes in $\mathcal{N} \setminus \{s, t\}$ to each class, where the first n_1 nodes are assigned to $\mathcal{N}_1$, and the remaining n_2 nodes are assigned to $\mathcal{N}_2$. To evaluate the performance of the approaches with respect to the density of the system, we generated three RGGs by setting the value of the parameter d to 1.5, 3.0, and 4.5. An example of the RGG generated with $d = 1.5$ is shown in Figure 5.

To conclude the experiments, we considered a realistic, large-scale power system. The 2000-bus power system, which has 4000 nodes and 29336 arcs and includes cycles and self-loops, is a synthetic but realistic electric grid model maintained by Texas A&M Smart Grid center (electricgrids.engr.tamu.edu); see (Birchfield et al., 2017a,b, 2018) for more information. We adapted the network by partitioning the nodes into two classes such that the first n_1 even-numbered nodes are assigned to class 1 and the remaining nodes are assigned to the second class.

4.3 Experiments Results

For the first set of experiments, we estimated the survival signatures of the Arpanet system and the Airplane electrical system for $\mathsf{M} = 100, 500, 1000, 5000, 10000$ replications. We then recorded the time each algorithm needs to estimate the survival signature and divide the respective time by the corresponding number of replications to obtain the rates (in seconds per replication, with four decimal digits) shown in Tables 4 and 5.


(a) System composed of three replicas of 1973 Arpanet.

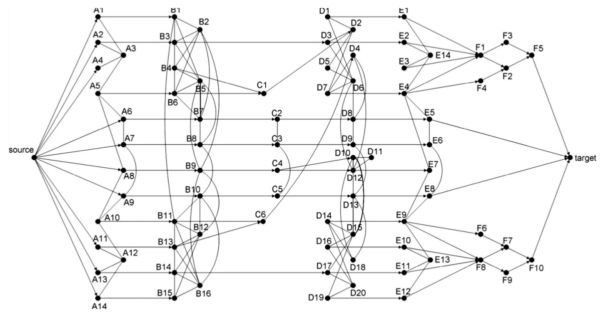

(b) Typical airplane electrical system.

Figure 4: Arpanet network and airplane electrical system adapted from Sebastio et al. (2014).

From Tables 4 and 5, we observe that the rate of each algorithm is approximately constant with a varying number of replications, and that their run time increases approximately linearly with an increase in M. This implies that we can obtain a good estimate of the total run time of an algorithm with respect to a system for a large M by running a small number of iterations, obtaining the rate, and multiplying it by M. Tables 4 and 5 also show the superiority of BO over the other three methods in these experiments. Between IS and SO, neither seems to dominate the other, while Naive was at least one order of magnitude slower than the other algorithms.

Table 4: Arpanet System with $n = 59$, $n_1 = 28$, $n_2 = 29$, $m = 142$.

All entries are given in seconds per replication					
	$\mathsf{M} = 100$	$\mathsf{M} = 500$	$\mathsf{M} = 1000$	$\mathsf{M} = 5000$	$\mathsf{M} = 10000$
Naive	0.0061	0.0061	0.0061	0.0061	0.0061
IS	0.0004	0.0004	0.0004	0.0004	0.0004
SO	0.0005	0.0005	0.0005	0.0005	0.0005
BO	**0.0002**	**0.0002**	**0.0002**	**0.0002**	**0.0002**

Table 5: Airplane electrical system with $n = 82$, $n_1 = 40$, $n_2 = 40$, $m = 268$.

All entries are given in seconds per replication					
	$\mathsf{M} = 100$	$\mathsf{M} = 500$	$\mathsf{M} = 1000$	$\mathsf{M} = 5000$	$\mathsf{M} = 10000$
Naive	0.0265	0.0264	0.0264	0.0265	0.0275
IS	0.0013	0.0012	0.0012	0.0012	0.0012
SO	0.0012	0.0011	0.0011	0.0011	0.0011
BO	**0.0005**	**0.0005**	**0.0005**	**0.0005**	**0.0005**

We start the second set of experiments using the same networks of the first set, but now we evaluate the effect of varying the size of the classes of nodes, $\mathcal{N}_1$ and $\mathcal{N}_2$, on the running time. For this set of experiments, we fix M at 5000 replications and estimate the survival signature for different combinations of n_1/n_2. For the Arpanet system, we consider the following combinations of n_1/n_2: 2/55, 5/52, 12/45, 20/37, and 28/29. For the Airplane system, we consider the following combinations of n_1/n_2: 2/78, 10/70, 20/60, 30/50, and 40/40. The results shown in Tables 6 and 7 reaffirm BO as the fastest method in all cases except for the extreme case where $n_1 = 2$, in which case SO seems to perform slightly better. These results also suggest that BO is the most robust against variations in the ratio n_1/n_2.

Table 6: Arpanet system for fixed $\mathsf{M} = 5000$ and different values of n_1 and n_2.

All entries are given in seconds per replication					
	$n_1 = 2,\ n_2 = 55$	$n_1 = 5,\ n_2 = 52$	$n_1 = 12,\ n_2 = 45$	$n_1 = 20,\ n_2 = 37$	$n_1 = 28,\ n_2 = 29$
Naive	0.0012	0.0022	0.0042	0.0056	0.0061
IS	0.0001	0.0001	0.0003	0.0004	0.0004
SO	0.0001	0.0001	0.0003	0.0004	0.0005
BO	0.0002	0.0001	**0.0002**	**0.0002**	**0.0002**

Table 7: Airplane system for fixed $\mathsf{M} = 5000$ and different values of n_1 and n_2.

All entries are given in seconds per replication					
	$n_1 = 2,\ n_2 = 78$	$n_1 = 10,\ n_2 = 70$	$n_1 = 20,\ n_2 = 60$	$n_1 = 30,\ n_2 = 50$	$n_1 = 40,\ n_2 = 40$
Naive	0.0037	0.0126	0.0204	0.0245	0.0265
IS	0.0002	0.0005	0.0008	0.0010	0.0012
SO	**0.0001**	0.0004	0.0007	0.0009	0.0011
BO	0.0003	**0.0003**	**0.0005**	**0.0005**	**0.0005**

To further investigate the performance and the robustness of the algorithms, we estimated the survival signature of the RGGs with $\mathsf{M} = 5000$ replications and varying values of n_1/n_2. Figure 5 shows the RGG generated for $d = 1.5$, $n_1 = 149$ and $n_2 = 199$.

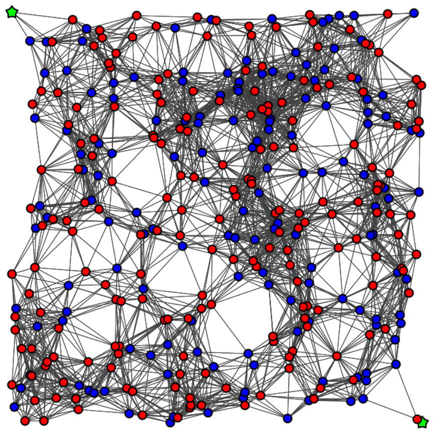


Figure 5: Network with $n = 350$, where $n_1 = 149$, $n_2 = 199$, and $m = 7446$ arcs.

Table 8 shows the results of the computational experiments performed with this RGG. For this RGG, the superiority of BO is highlighted in terms of both running time and robustness in almost all scenarios. The exception is again the extreme case of $n_1 = 2$. In terms of run time rates, BO becomes orders of magnitude faster than the other approaches in most cases with SO competing an order of magnitude slower than BO in most cases. The BO approach is also the most robust when we increase n_1 as shown in Table 8.

Table 8: Random Geometric Graph system with 350 nodes with $d = 1.5$ and different values of n_1 and n_2.

All entries are given in seconds per replication RGG for $d = 1.5$, $m = 7446$, density=0.06				
	$n_1 = 2,\ n_2 = 346$	$n_1 = 49,\ n_2 = 299$	$n_1 = 99,\ n_2 = 249$	$n_1 = 149,\ n_2 = 199$
Naive	0.2406	4.1139	8.5229	9.5301
IS	0.0103	0.1556	0.2491	0.4188
SO	**0.0016**	0.0219	0.0376	0.0508
BO	0.0037	**0.0056**	**0.0075**	**0.0079**

Notice that the density of the RGG discussed above (created with $d = 1.5$) is approximately 0.06. For $d = 3.0$, the resulting RGG has 25412 arcs and a density of 0.21. Lastly, for $d = 4.5$ the corresponding RGG has 49208 arcs and a density of 0.40. The results for the RGGs with $d = 3.0$ and $d = 4.5$ are shown in Tables 9 and 10.

Tables 8, 9 and 10 reinforce the dominance of BO over the benchmarks for practical cases and highlight

Table 9: Random Geometric Graph system with 350 nodes with $d = 3.0$ and different values of n_1 and n_2.

All entries are given in seconds per replication RGG for $d = 3.0$, $m = 25412$, density=0.21				
	$n_1 = 2,\ n_2 = 346$	$n_1 = 49,\ n_2 = 299$	$n_1 = 99,\ n_2 = 249$	$n_1 = 149,\ n_2 = 199$
Naive	0.4151	8.0551	11.2330	12.9170
IS	0.1130	1.7651	2.6720	4.0805
SO	**0.0027**	0.0542	0.0740	0.0988
BO	0.0095	**0.0111**	**0.0125**	**0.0130**

Table 10: Random Geometric Graph system with 350 nodes with $d = 4.5$ and different values of n_1 and n_2.

All entries are given in seconds per replication RGG for $d = 4.5$, $m = 49208$, density=0.40				
	$n_1 = 2,\ n_2 = 346$	$n_1 = 49,\ n_2 = 299$	$n_1 = 99,\ n_2 = 249$	$n_1 = 149,\ n_2 = 199$
Naive	0.7278	11.2240	15.2810	20.6890
IS	0.5217	7.4703	12.0010	12.0300
SO	**0.0041**	0.06154	0.1110	0.1544
BO	0.0174	**0.0178**	**0.0180**	**0.0181**

its robustness against variations in n_1 and network density. In these experiments, SO outperforms the other approaches for $n_1 = 2, n_2 = 346$, and is markedly faster than IS, whereas in the previous experiments this difference was not clear.

Lastly, using BO, SO, and IS, we estimated the survival signature of the 2000-bus power system with M = 250 replications for varying values of n_1/n_2. Because Naive required in excess of 10 hours per replication, we excluded it from this experiment. Table 11 shows the results for the other three approaches.

Table 11: TAMU SmartGridCenter 2000-bus power system.

All entries are given in seconds per replication				
	$n_1 = 2,\ n_2 = 3996$	$n_1 = 999,\ n_2 = 2999$	$n_1 = 1499,\ n_2 = 2499$	$n_1 = 1999,\ n_2 = 1999$
IS	**0.0218**	5.3674	5.7789	6.3404
SO	0.1235	26.1520	36.1740	44.0640
BO	0.1012	**0.1872**	**0.1851**	**0.2040**

Table 11 confirms the results of previous experiments and highlights the advantages of BO over the other benchmark methods in almost all scenarios. BO algorithm scaled well and in the worst case ($n_1 = 1999, n_2 = 1999$) yielded a rate of 0.2040 seconds per iteration, demonstrating that BO is well-suited to handle large-scale systems. For instance, for the same setting but changing M to a more realistic number such as 10000 replications, BO would complete the survival signature estimation in less than one hour. By comparison, IS yielded a rate of 6.34 seconds per iteration in the worst case and would take more than 17 hours to run 10000 replications.

As a secondary point, it is interesting to note that in the RGG experiments (Tables 8, 9 and 10), SO outperformed IS by a considerable margin, whereas in the 2000-bus power system experiments, SO fared markedly worse than IS even for the extreme case where $n_1 = 2, n_2 = 3996$. Although in both cases the network considered was a large-scale system, the density in all three RGGs were considerably larger than the density of the 2000-bus power system, which is approximately 0.0018. This result suggests that SO may not perform well with massive but sparse networks compared to IS. Although SO and IS are equal in worst-case complexity, we believe this difference in performance can be explained by differences in the computations carried out by each algorithm. In SO, each arc is considered at most once but some additional work is created due to manipulating the bucket data structures. On the other hand, IS may consider each arc at most twice: once when scanning the predecessor list of the arc's head node and once when scanning the successor list of its tail node. Furthermore, in instances where s and t become connected after adding only a few nodes, it is possible for a majority of the arcs considered when scanning predecessor lists to be arcs that are never encountered when scanning successor lists, meaning the number of arcs examined in IS can be more than twice the number examined in SO. This difference in computational effort can become more pronounced as the number of arcs grows, thus leading SO to be preferred for denser networks. We did not pursue further investigation in this direction since BO decisively outperformed both SO and IS for every instance of considerable size and in all scenarios except for the extreme cases where $n_1 = 2$.

5 Considerations and Future Research

In this paper, we proposed a bi-objective optimization-based MC method (which we refer to as BO) to estimate the two-terminal survival signature of networks with two component classes. To the best of our knowledge, this is the first work to point out the relationship between survival signature computation and multi-objective optimization. In addition, we discussed three alternative approaches to estimate the two-terminal survival signature within a MC framework without exploiting the relation to multi-objective optimization.

We conducted extensive computational experiments to compare the performance of the methods in terms of run time and robustness. Although IS and SO have better worst-case complexity, the experiments revealed

BO to be the fastest and the most robust method. The experiments also showed that BO is well-suited to estimate the survival signature of realistic, large-scale systems, while the other three approaches become unpractical as the size of the network becomes large.

This work contributes to the network reliability literature from both theoretical and practical perspectives. From a theoretical perspective, the possibility of estimating the survival signature by solving a multi-objective network optimization problem, and the efficiencies gained by doing so, indicates an interesting path for developing efficient algorithms for reliability estimation. From an practical perspective, we have shown that BO is well-suited to estimate the survival signature of realistic, generally structured systems in a reasonable amount of time even for systems with thousands of nodes and arcs.

Interesting future research directions branching from this work includes: (i) generalizing our approach to estimate the two-terminal survival signature for networks with more than two component classes; (ii) generalizing our approach to multi-state systems; and (iii) generalizing our approach to other reliability metrics such as K-terminal reliability and coverage metrics.

6 Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Number CMMI-1751191. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

J. L. Cook and J. E. Ramirez-Marquez. Two-terminal reliability analyses for a mobile ad hoc wireless network. *Reliability Engineering & System Safety*, 92(6):821–829, 2007.

B. A. Gebre and J. E. Ramirez-Marquez. Element substitution algorithm for general two-terminal network reliability analyses. *IIE Transactions*, 39(3):265–275, 2007.

J. Silva, T. Gomes, D. Tipper, and L. Martins. An effective algorithm for computing all-terminal reliability bounds. *Networks*, 66(4):282–295, 2015.

S. Chakraborty, N. K. Goyal, S. Mahapatra, and S. Soh. Minimal path-based reliability model for wireless sensor networks with multistate nodes. *IEEE Transactions on Reliability*, 69(1):382–400, 2020.

L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3): 410–421, 1979.

J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.

M. O. Ball. Computational complexity of network reliability analysis: An overview. *IEEE Transactions on Reliability*, 35(3):230–239, 1986.

F. J. Samaniego. On closure of the IFR class under formation of coherent systems. *IEEE Transactions on Reliability*, R-34:69–72, 1985.

I. Gertsbakh and Y. Shpungin. *Models of Network Reliability: analysis, combinatorics, and Monte Carlo*. CRC Press, Boca Raton, 2009.

F. P. A. Coolen and T. Coolen-Maturi. Generalizing the signature to systems with multiple types of components. In W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, and J. Kacprzyk, editors, *Complex Systems and Dependability*, pages 115–130, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

I. B. Gertsbakh, Y. Shpungin, and R. Vaisman. D-spectrum and reliability of a binary system with ternary components. *Probability in the Engineering and Informational Sciences*, 30:25–39, 2016.

T. Elperin, I. Gertsbakh, and M. Lomonosov. Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, 40(5):572–581, 1991.

A. Sedeño-Noda and M. Colebrook. A biobjective Dijkastra algorithm. *European Journal of Operational Research*, 276:106–118, 2019.

E. F. Moore and C. E. Shannon. Reliable circuits using less reliable relays. *Journal of Franklin Institute*, 1956.

R. E. Barlow and F. Proschan. *Mathematical Theory of Reliability*. John Wiley & Sons, 1965.

I. Brown, C. Graves, B. Miller, and T. Russell. Most reliable two-terminal graphs with node failures. *Networks*, 76:414–426, 2020.

C.-C. Jane and J. Yuan. A sum of disjoint products algorithm for reliability evaluation of flow networks. *European Journal of Operational Research*, 131:664–675, 2001.

E. Datta and N. K. Goyal. Sum of disjoint product approach for reliability evaluation of stochastic flow networks. *International Journal of Systems Assurance Engineering and Management*, 8:1734–1749, 2017.

T. Aven. Reliability evaluation of multistate systems with multistate components. *IEEE Transactions on Reliability*, R-34(5):473–479, 1985.

C. Alexopoulos. A note on state-space decomposition methods for analyzing stochastic flow networks. *IEEE Transactions on Reliability*, 44(2):354–357, 1995.

G. Bai, Z. Tian, and M. J. Zuo. Reliability evaluation of multistate networks: An improved algorithm using state-space decomposition and experimental comparison. *IISE Transactions*, 50(5):407–418, 2018.

J. E. Ramirez-Marquez, D. Coit, and M. Tortorella. A generalized multi-state-based path vector approach to multistate two-terminal reliability. *IIE Transactions*, 38(6):477–488, 2006.

F. Moskowitz. The analysis of redundancy networks. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, 77(5):627–632, 1958.

A. Satyanarayana and M. K. Chang. Network reliability and the factoring theorem. *Networks*, 13(1):107–120, 1983.

R. K. Wood. A factoring algorithm using polygon-to-chain reductions for computing k-terminal network reliability. *Networks*, 15(2):173–190, 1985.

R. K. Wood. Factoring algorithms for computing k-terminal network reliability. *IEEE Transactions on Reliability*, 35(3):269–278, 1986.

J. M. Burgos and F. R. Amoza. Factorization of network reliability with perfect nodes I: Introduction and statements. *Discrete Applied Mathematics*, 198:82–90, 2016.

G. Hardy, C. Lucet, and N. Limnios. K-terminal network reliability measures with binary decision diagrams. *IEEE Transactions on Reliability*, 56(3):506–515, 2007.

S.-Y. Kuo, F.-M. Yeh, and H.-Y. Lin. Efficient and exact reliability evaluation for networks with imperfect vertices. *IEEE Transactions on Reliability*, 56(2):288–300, 2007.

C.-C. Jane, W.-H. Shen, and Y.-W. Laih. Practical sequential bounds for approximating two-terminal reliability. *European Journal of Operational Research*, 195:427–441, 2009.

M. Lê, M. Walter, and J. Weidendorfer. A memory-efficient bounding algorithm for the two-terminal reliability problem. *Electronic Notes in Theoretical Computer Science*, 291:15–25, 2013.

S. Sebastio, K. S. Trivedi, D. Wang, and X. Yin. Fast computation of bounds for two-terminal network reliability. *European Journal of Operational Research*, 238:810–823, 2014.

C. Srivaree-ratana, A. Konak, and A. E. Smith. Estimation of all-terminal network relibility using an artificial neural network. *Computers & Operations Research*, 29:849–868, 2002.

F. Altiparmak, B. Dengiz, and A. E. Smith. A general neural network model for estimating telecommunications network reliability. *IEEE Transactions on Reliability*, 58(1):2–9, 2009.

Z. Zhang and F. Shao. A diameter-constrained approximation algorithm of multistate two-terminal reliability. *IEEE Transactions on Reliability*, 67(3):1249–1260, 2018.

K.-P. Hui, N. Bean, M. Kraetzl, and D. P. Kroese. The cross-entropy method for network reliability estimation. *Annals of Operations Research*, 134:101–118, 2005.

A. Heidarzadeh, A. Sprintson, and C. Singh. A fast and accurate failure frequency approximation for k-terminal reliability systems. *IEEE Transactions on Reliability*, 67(3):933–950, 2018.

G. Cristescu and V.-F. Dragoi. Efficient approximation of two-terminal networks reliability polynomials using cubic splines. *IEEE Transactions on Reliability*, 70(3):1193–1203, 2021.

G. S. Fishman. A comparison of four Monte Carlo methods for estimating the probability of s-t connectedness. *IEEE Transactions on Reliability*, 35(2):145–155, 1986.

J. E. Ramirez-Marquez and B. A. Gebre. A classification tree based approach for the development of minimal cut and path vectors of a capacitated network. *IEEE Transactions on Reliability*, 56(3):474–487, 2007.

R. E. Stern, J. Song, and D. B. Work. Accelerated Monte Carlo system reliability analysis through machine-learning-based surrogate models of network connectivity. *Reliability Engineering & System Safety*, 164: 1–9, 2017.

S. Reed, M. Löfstrand, and J. Andrews. An efficient algorithm for computing exact system and survival signatures of k-terminal network reliability. *Reliability Engineering & System Safety*, 185:429–439, 2019.

J. Behrensdorf, T.-E. Regenhardt, M. Broggi, and M. Beer. Numerically efficient computation of the survival signature for the reliability analysis of large networks. *Reliability Engineering & System Safety*, 216:107935, 2021.

J. Navarro, F. J. Samaniego, N. Balakrishnan, and D. Bhattacharya. On the application and extension of system signatures in engineering reliability. *Naval Research Logistics*, 55:317–327, 2008.

J. Navarro, F. J. Samaniego, and N. Balakrishnan. Signature-based representations for the reliability of systems with heterogeneous components. *Journal of Applied Probability*, 48(3):856–867, 2011.

I. Gertsbakh and Y. Shpungin. *Network Reliability and Resilience*. Springer Briefs in Electrical and Computer Engineering. Springer, 2011.

I. B. Gertsbakh, Y. Shpungin, and R. Vaisman. Reliability of a network with heterogeneous components. In A. Lisnianski, L. Frenkel, and A. Karagrigoriou, editors, *Recent Advances in Multi-state Systems Reliability: Theory and Applications*, Springer Series in Reliability Engineering, pages 3–18. Springer, 2018.

F. J. Samaniego and J. Navarro. On comparing coherent systems with heterogeneous components. *Advances in Applied Probability*, 48(1):88–111, 2016.

S. Kochar, H. Mukerjee, and F. J. Samaniego. The "signature" of a coherent system and its application to comparisons among systems. *Naval Research Logistics*, 46:507–523, 1999.

Y. Shpungin. Networks with unreliable nodes and edges: Monte Carlo lifetime estimation. *International Journal of Electrical and Computer Engineering*, 1(3):466–471, 2007.

A. M. Andronov, I. B. Gertsbakh, and Y. Shpungin. On an application of signatures (D-Spectra) to analysis of single-line queueing system. *Automatic Control and Computer Sciences*, 45(4):181–191, 2011.

F. J. Samaniego, N. Balakrishnan, and J. Navarro. Dynamic signatures and their use in comparing the reliability of new and used systems. *Naval Research Logistics*, 56:577–591, 2009.

I. B. Gertsbakh and Y. Shpungin. Failure development in a system of two connected networks. *Transport and Telecommunication*, 13(4):255–260, 2012.

B. H. Lindqvist and F. J. Samaniego. On the signature of a system under minimal repair. *Applied Stochastic Models in Business and Industry*, 31:297–306, 2015.

F. P. A. Coolen, T. Coolen-Maturi, and A. H. Al-Nefaiee. Nonparametric predictive inference for system reliability using the survival signature. *Journal of Risk and Reliability*, 228(5):437–448, 2014.

L. J. M. Aslett, F. P. A. Coolen, and S. P. Wilson. Bayesian inference for reliability of systems and networks using the survival signature. *Risk Analysis*, 35(9), 2015.

A. Najem and F. P. A. Coolen. System reliability and component importance when components can be swapped upon failure. *Applied Stochastic Models in Business and Industry*, 35:399–413, 2018.

S. Eryilmaz, F. P. A. Coolen, and T. Coolen-Maturi. Mean residual life of coherent systems consisting of multiple types of dependent components. *Naval Research Logistics*, 65:86–97, 2018.

T. Coolen-Maturi, F. P. A. Coolen, and N. Balakrishnan. The joint survival signature of coherent systems with shared components. *Reliability Engineering & System Safety*, 207:107350, 2021.

S. Eryilmaz and A. Tuncel. Generalizing the survival signature to unreparable homogeneous multi-state systems. *Naval Research Logistics*, 63(8):593–599, 2016.

H. Yi, N. Balakrishnan, and X. Li. Signatures of multi-state systems based on a series/parallel/recurrent structure of modules. *Probability in the Engineering and Informational Sciences*, 36(3):824–850, 2022.

S. Reed. An efficient algorithm for exact computation of system and survival signatures using binary decision diagrams. *Reliability Engineering & System Safety*, 165:257–267, 2017.

B.-H. Xu, D.-Z. Yang, C. Qian, Q. Feng, Y. Ren, Z.-L. Wang, and B. Sun. A new method for computing survival signature based on extended universal generating function. In *2019 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)*, 2019.

F. Di Maio, C. Pettorossi, and E. Zio. Entropy-driven monte carlo simulation method for approximating the survival signature of complex infrastructures. *Reliability Engineering & System Safety*, 231:108982, 2023.

E. Patelli, G. Feng, F. P. A. Coolen, and T. Coolen-Maturi. Simulation methods for system reliability using survival signature. *Reliability Engineering & System Safety*, 167:327–337, 2017.

N. T. Boardman and K. M. Sullivan. Time-based node deployment policies for reliable wireless sensor networks. *IEEE Transactions on Reliability*, 70(3):1204–1217, 2021.

D. Ferone, P. Festa, S. Fugaro, and T. Pastore. A dynamic programming algorithm for solving the k-color shortest path problem. *Optimization Letters*, 15(6):1973–1992, 2021.

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Inc., 1993.

M. Pollack. Letter to the editor - the maximum capacity through a network. *Operations Research*, 8(5): 733–736, 1960.

T. C. Hu. Letters to the editor - the maximum capacity route problem. *Operations Research*, 9(6):898–900, 1961.

A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye. Grid structural characteristics as validation criteria for synthetic networks. *IEEE Transactions on Power Systems*, 32(4):3258–3265, 2017a.

A. Birchfield, E. Schweitzer, M. Athari, T. Xu, T. Overbye, A. Scaglione, and Z. Wang. A metric-based validation process to assess the realism of synthetic power grids. *Energies*, 10(8):1233, 2017b.

A. B. Birchfield, T. Xu, and T. J. Overbye. Power flow convergence and reactive power planning in the creation of large synthetic grids. *IEEE Transactions on Power Systems*, 33(6):6667–6674, 2018.

Appendix A Naive Approach Algorithm

Algorithm 5: Naive

```
Φ ← 0
for $j = 1$ to M do
    ▷ Generate a permutation of $\mathcal{N}_1$ and a permutation of $\mathcal{N}_2$
    ▷ Represent the permutations according to Equation (7)
    ▷ Turn DOWN $q_i^1$ for $i = 1, \ldots, n_1$
    for $\ell_1 = 0$ to $n_1$ do
        ▷ Turn DOWN $q_i^2$, for $i = 1, \ldots, n_2$
        for $\ell_2 = 0$ to $n_2$ do
            ▷ Run a BFS from node $s$
            if the BFS reaches $t$ then
                $\phi(\ell_1, \ell_2) \leftarrow \phi(\ell_1, \ell_2) + 1$
            end
            if $\ell_2 < n_2$ then
                ▷ Turn $q^2_{n_2-\ell_2}$ UP
            end
        end
        if $\ell_1 < n_1$ then
            ▷ Turn $q^1_{n_1-\ell_1}$ UP
        end
    end
    $\Phi \leftarrow \frac{1}{\mathsf{M}}\Phi$
end
return $\Phi$
```

Appendix B Single-objective Optimization Approach Algorithms

Algorithm 6: Dials-MaxCapPath

```
bucket$[i] \leftarrow \varnothing,\ \forall i \in \{0, 1, \ldots, n\}$
d$(i) \leftarrow 0,\ \forall\ i \in \mathcal{N} \setminus \{s\}$
d$(s) \leftarrow n$
$idx \leftarrow n$
bucket$[idx] \leftarrow$ bucket$[idx] \cup \{s\}$
while $idx \geq 0$ do
    if bucket$[idx] \neq \varnothing$ then
        for $i \in$ bucket$[idx]$ do
            for $j \in \Gamma_i^+$ do
                if $d(j) < \min\{d(i), u_j\}$ then
                    if $d(j) > 0$ then
                        bucket$[d(j)] \leftarrow$ bucket$[d(j)] \setminus \{j\}$
                    end
                    $d(j) \leftarrow \min\{d(i), u_j\}$
                    bucket$[d(j)] \leftarrow$ bucket$[d(j)] \cup \{j\}$
                end
            end
        end
    end
    $idx \leftarrow idx - 1$
end
return d$(t)$
```

Algorithm 7: SO

```
$\Phi \leftarrow \mathbf{0}$
for $j = 1$ to M do
    $\triangleright$ Simulate a permutation of $\mathcal{N}_1$ and a permutation of $\mathcal{N}_2$
    for $\ell_1 = 0$ to $n_1$ do
        $\triangleright$ Update $u_i,\ i \in \mathcal{N}$, according to Equation (10)
        $v^* \leftarrow$ SO-MaxCapPath$(\mathcal{G})$
        ${\ell_2}^* \leftarrow n_2 - v^*$
        for $\ell_2 = 0$ to $n_2$ do
            if $\ell_2 > {\ell_2}^*$ then
                $\phi(\ell_1, \ell_2) \leftarrow \phi(\ell_1, \ell_2) + 1$
            end
        end
    end
end
$\Phi \leftarrow \Phi/$M.

return $\Phi$
```

Appendix C Single-objective Optimization Approach Complexity

Proposition C.1. *The overall complexity of the* SO *approach is* $O(n_1 n\mathsf{M} + n_1 m\mathsf{M})$.

Proof. We start by stating the complexity of Algorithm 6. The initialization portion of the algorithm (steps 1–5) can be performed in $O(n)$, but the dominating factor is the while loop which is used by the algorithm to check buckets and update labels. Using doubly linked list as the underlying structure of the bucket sets, each label update is performed in $O(1)$ and since each label is made permanent once, we examine each arc at most once, and the total work through the algorithm to update all labels (lines 9–16) requires $O(m)$ time. Checking all the bucket sets can be performed in $O(n)$ since the algorithm keeps $n + 1$ buckets and it checks each bucket once. Thus, the complexity of Algorithm 6 is $O(n + m)$. Since Algorithm 7 solves a MCP for each value of ℓ_1 and MC replication, the overall complexity of SO is $O(n_1 n\mathsf{M} + n_1 m\mathsf{M})$, which is equal to $O(n_1 m\mathsf{M})$ provided that $n \leq m$. □

Appendix D Complexity of Bi-objective Optimization Approach

Proposition D.1. *The complexity of Algorithm 1 is $O(n_1^2 m + n_1 m \log n)$.*

Proof. We first analyze the work performed for every iteration of the while loop. When a node label l^* is made permanent, the algorithm performs a $\mathsf{find\text{-}max}$ operation in $O(1)$, a $\mathsf{delete\text{-}max}$ operation in $O(\log n)$, a series of assignment operations each having complexity $O(1)$, and, when the label returned by $\mathsf{NewCandidateLabel}$ is not Null, the algorithm performs an insert in $O(\log n)$ and another series of assignment operations in constant time. The work performed in one iteration of the while loop is therefore $O(\log n)$ plus the complexity of $\mathsf{NewCandidateLabel}$ and $\mathsf{RelaxationProcess}$. The algorithm performs an iteration of the while loop N_i times for every $i \in \mathcal{N}$, and since $\mathrm{N}_i \leq n_1 + 1$ (because every non-dominated point must have a different value of d_i^1), the work performed for all nodes is $O(n_1 n \log n)$ plus the complexity of $\mathsf{NewCandidateLabel}$ and $\mathsf{RelaxationProcess}$. In every iteration, function $\mathsf{NewCandidateLabel}$ performs a series of comparisons and assignments in constant time $|\mathsf{L}[j]| \leq \mathrm{N}_j$ times for every $j \in \Gamma_i^-$. Because $\mathsf{NewCandidateLabel}$ is called exactly N_i times for each node (i.e., once for every label made permanent), the total work for node i is $O(\mathrm{N}_i \sum_{j\in\Gamma_i^-} \mathrm{N}_j) = O(n_1 \sum_{j\in\Gamma_i^-} n_1) = O(n_1^2 |\Gamma_i^-|)$. Therefore, the total time due to $\mathsf{NewCandidateLabel}$ is $O(\sum_{i\in\mathcal{N}} n_1^2 |\Gamma_i^-|) = O(n_1^2 \sum_{i\in\mathcal{N}} |\Gamma_i^-|) = O(n_1^2 m)$, noting that $\sum_{i\in\mathcal{N}} |\Gamma_i^-| = m$.

In each iteration, for every $j \in \Gamma_i^+$, $\mathsf{RelaxationProcess}$ performs a sequence of constant time operations and either an insert or $\mathsf{increase\text{-}key}$ operation in the worst case with complexity $O(\log n)$. Like the $\mathsf{NewCandidateLabel}$, $\mathsf{RelaxationProcess}$ is called N_i times for each $i \in \mathcal{N}$, which amounts to a total complexity of $O(\sum_{i\in\mathcal{N}} (\mathrm{N}_i \log n |\Gamma_i^+|))$. Noting that $\mathsf{N}_i = O(n_1)$ and $\sum_{i\in\mathcal{N}} |\Gamma_i^+| = m$, this complexity reduces to $O(n_1 m \log n)$. Comparing the results above, the overall complexity of Algorithm 1 is $O(n n_1 \log n + n_1^2 m + n_1 m \log n)$, which is $O(n_1^2 m + n_1 m \log n)$ provided that $n < m$. $\square$

Appendix E Validation and Accuracy of the Proposed Methods

We assess the accuracy of our methods by considering a *chain* network constructed as follows: given a positive integer number n_1, we have nodes $0, 1, 2, \ldots, 3n_1$, where $\mathcal{N}_1 = \{1, 4, 7, \ldots, 3n_1-2\}$, $\mathcal{N}_2 = \{2, 5, 8, \ldots, 3n_1-1\}$, and nodes $\{0, 3, 6, \ldots, 3n_1\}$ are completely reliable. The arcs are created as follows: let $i = 0, 3, 6, \ldots, 3n_1$, add the arcs $(i, i+1)$, $(i, i+2)$, $(i+1, i+3)$, and $(i+2, i+3)$. Nodes 0 and $3n_1$ are, respectively, the source and sink nodes in this network.

The example below shows a chain network for $n_1 = 8$, where $\mathcal{N}_1 = \{1, 3, 7, 10, 13, 16, 19, 22\}$, $\mathcal{N}_2 = \{2, 5, 8, 11, 14, 17, 20, 23\}$, and $\{s, 3, 6, 9, 12, 15, 18, 21, t\}$ are completely reliable nodes.

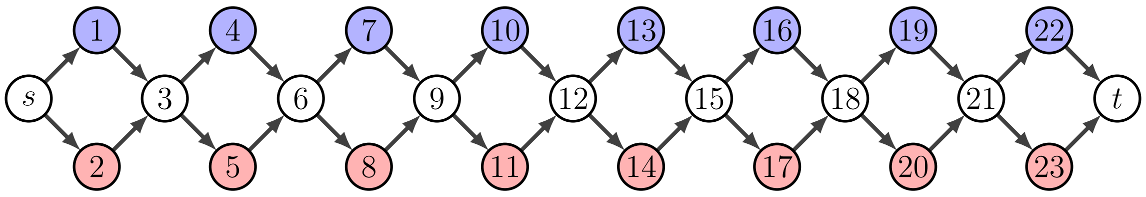


Figure 6: *Chain* network with $n_1 = 8$.

We can compute the exact survival signature of this network for a positive integer parameter n_1 denoting the number of nodes in each class ($\mathcal{N}_1$ and $\mathcal{N}_2$) by making the following observations: (i) at least n_1 UP components from $\mathcal{N}_1 \cup \mathcal{N}_2$ are necessary to have s-t connection; therefore, $\phi(\ell_1, \ell_2) = 0$ for any entry such that $\ell_1 + \ell_2 < n_1$; (ii) if all n_1 nodes from the same class are UP, the network is UP: $\phi(\ell_1, \ell_2) = 1$ for any entry such that $\ell_1 = n_1$ or $\ell_2 = n_1$; and (iii) for $\ell_1 + \ell_2 \geq n_1$, we have the following result.

Proposition E.1. *Given a* chain *network constructed according to the procedure described above, the exact survival signature entries for $\ell_1 + \ell_2 \geq n_1$ can be computed according to*

$$\phi(\ell_1, \ell_2) = \frac{\binom{\ell_1}{\ell_2-(n_1-\ell_1)}}{\binom{n_1}{\ell_2}}. \tag{16}$$

Proof. Given a subset $Q \subseteq \mathcal{N}_1$ of ℓ_1 UP nodes from $\mathcal{N}_1$, let $R \subseteq \mathcal{N}_2$ denote the set of $n_1 - \ell_1$ nodes from $\mathcal{N}_2$ that must be UP for the network to be UP. Since each subset of ℓ_2 nodes in $\mathcal{N}_2$ is equally likely to comprise the set of UP nodes for $\mathcal{N}_2$, the network is UP if a sample of ℓ_2 nodes from $\mathcal{N}_2$ (with nodes drawn equally likely and without replacement) contains all the nodes in R. Thus, the probability that network is UP given that the ℓ_1 nodes in Q are UP follows a hypergeometric distribution given by the number of samples of ℓ_2 nodes in $\mathcal{N}_2$ that contain all of the nodes in R divided by the number of samples of ℓ_2 nodes in $\mathcal{N}_2$. Because the sample in the numerator must include all of the nodes in R, we can rewrite the numerator as the number of samples of $\ell_2 - (n_1 - \ell_1)$ nodes in $\mathcal{N}_2 \setminus R$, which equals $\binom{\ell_1}{\ell_2-(n_1-\ell_1)}$. $\square$

Table 12 shows the exact survival signature for the network of Figure 16 up to 7 decimal digits, and Table 13 shows the estimates obtained by applying our method with $M = 20000$ replications.

Table 12: Exact survival signature (up to 7 decimal digits).

$\ell_1 \backslash \ell_2$	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0.125	1
2	0	0	0	0	0	0	0.0357143	0.25	1
3	0	0	0	0	0	0.0178571	0.107143	0.375	1
4	0	0	0	0	0.0142857	0.0714286	0.214286	0.5	1
5	0	0	0	0.0178571	0.0714286	0.178571	0.357143	0.625	1
6	0	0	0.0357143	0.107143	0.214286	0.357143	0.535714	0.75	1
7	0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1
8	1	1	1	1	1	1	1	1	1

Table 13: Estimated survival signature ($M = 20000$ replications).

$\ell_1 \backslash \ell_2$	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0.1244	1
2	0	0	0	0	0	0	0.03525	0.25265	1
3	0	0	0	0	0	0.0178	0.10595	0.37195	1
4	0	0	0	0	0.01585	0.0707	0.21495	0.50325	1
5	0	0	0	0.0193	0.0697	0.17685	0.35525	0.6261	1
6	0	0	0.03505	0.1056	0.2135	0.3567	0.53645	0.75035	1
7	0	0.12225	0.24335	0.36905	0.4971	0.62065	0.74775	0.87545	1
8	1	1	1	1	1	1	1	1	1

We evaluated the accuracy of the estimates by performing, for every entry whose exact signature is different from 0 and different from 1, the following hypothesis test on proportions:

H_0: The MC estimate $\hat{\phi}(\ell_1, \ell_2)$ equals the exact signature $\phi(\ell_1, \ell_2)$.

H_1: $\hat{\phi}(\ell_1, \ell_2) \neq \phi(\ell_1, \ell_2)$.

For a level of significance $\alpha = 0.05$, the test resulted in "reject" for only one of the estimates, which implies a 3.57% rejection rate when considering the 28 entries for which $0 < \phi(\ell_1, \ell_2) < 1$, and therefore very close to the 5% rate that is expected in this setting.

We also computed the exact and estimated ($M = 75000$ replications) signatures for a *chain* network with $n_1 = 20$, i.e., 20 nodes in $\mathcal{N}_1$, 20 nodes in $\mathcal{N}_2$, and 61 nodes in total. As before, we conducted the hypothesis test on proportion, which resulted in "reject" for 6 of the 190 entries with $0 < \phi(\ell_1, \ell_2) < 1$, indicating a 3.16% rejection rate and, therefore, very close to the 5% rate that is expected for $\alpha = 0.05$. All the files corresponding to the exact and estimated signatures, as well as an *R*-script with the hypothesis test and the files containing the *p*-values for each network instance can be found on the repository.