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

May 26, 2023

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; Beichelt and Tittmann, 2012; Silva et al., 2015; Caşcaval and Floria, 2017; 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

In this section, we summarize the relevant literature, beginning with a discussion of methodologies that have been used to evaluate two-terminal reliability either exactly or approximately. We then provide a brief review of signatures, their relationship, and their applications on reliability estimation and evaluation. We close the section by discussing other closely related works.

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; Caşcaval and Floria, 2017), state-space decomposition (Doulliez and Jamoulle, 1972; 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) (Lin et al., 2003; 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 upon 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 Coit (2005) propose a MC method to estimate the two-terminal reliability of a multi-state network, i.e., where the network components can be in multiple states of performance as opposed to (only) failed or functional. Their method employs MC simulation to generate system state vectors, which are then compared to the set of multi-state minimum cut vectors to determine whether the system state vector achieves the demand level required. Their method is specialized in (Cook and Ramirez-Marquez, 2007) to leverage the specific features of mobile ad hoc wireless networks.

Ramirez-Marquez and Gebre (2007) present an MC method to approximate bounds on the two-terminal reliability of capacitated networks. The method is based on simulating network configurations, evaluating these configurations using exact methods, generating corresponding classification trees, and obtaining minimal cut or path vectors from the classification trees.

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. Support vector machine and logistic regression based on surrogate models of node connectivity are applied within the MC method to reduce its runtime.

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

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

2.2.1 System Signature

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 $(i = 1, 2, \ldots, n)$ 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 computation of s_i is based on permutations of the components' failure times. Under the i.i.d. lifetimes assumption, the $n!$ permutations of $(1, 2, \ldots, n)$ are equally likely outcomes of the order in which components fail. Hence, we can (in theory) generate all $n!$ permutations and assess for each permutation how many failures are needed to cause the system to fail. Then, s_i is the proportion of permutations in which the ith failure causes the system's failure.

Given the system signature, the reliability of the system can be calculated by

$$P\{T > \tau\} = \sum_{i=1}^{n} s_i \sum_{j=0}^{i-1} \binom{n}{j} [F(\tau)]^j [1 - F(\tau)]^{n-j}, \tag{2}$$

where T is the system lifetime and $F(\tau)$ is the common lifetime CDF of components $i = 1, \ldots, n$ (Samaniego and Navarro, 2016). The system signature is closely related to the destruction spectrum (Gertsbakh and Shpungin, 2009, 2011; Gertsbakh et al., 2018) and has 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, 2007a), 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).

Shaked and Suarez-Llorens (2003) compute the signatures of all coherent systems with 2, 3, and 4 components. Navarro and Rubio (2009) present a minimal path set algorithm to obtain all n-component coherent systems and compute their signatures. Although their algorithm was applied to obtain the signatures of all 5-component coherent systems, the algorithm's run time grows exponentially and therefore cannot handle large systems. Da et al. (2012) and Da et al. (2018) propose methods for computing the signature of specially structured systems, e.g., systems composed of smaller disjoint subsystems or subsystems with shared components. Yi and Cui (2018) propose a Markov process method to compute the system signature of some coherent, consecutive-type systems, but they note that their method may not handle large, generally-structured systems.

For large, generally-structured systems, the combinatorial difficulty in obtaining the exact system signature/destruction spectrum has prompted researchers to focus on estimating methods based on MC simulation; see, for instance, Gertsbakh and Shpungin (2004); Shpungin (2007a,b); Gertsbakh et al. (2011, 2016).

2.2.2 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* $\phi(l)$, for $l = 0, \ldots, n$, 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 $\phi(l)$, $l = 0, \ldots, n$. Let S_l denote the set of state vectors in which $x_i = 1$ for exactly l components, and observe that $|S_l| = \binom{n}{l}$. Because the components are i.i.d., all vectors in S_l are equally likely, and therefore

$$\phi(l) = \binom{n}{l}^{-1} \sum_{\mathbf{x} \in S_l} \Psi(\mathbf{x}),\ \ l = 0, 1, \ldots, n, \tag{3}$$

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

$$\phi(l) = \sum_{j=n-l+1}^{n} s_j,\ \ l = 0, 1, \ldots, n. \tag{4}$$

The right-hand side of Equation (4) denotes the probability that at least $(n-l+1)$ 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 (4).

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 $\phi(l_1, l_2, \ldots, l_K)$ denote the probability that a system functions if exactly $l_k \in \{0, 1, \ldots, n_k\}$ 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 $\phi(l_1, l_2, \ldots, l_K)$ for all the values of $l_1, l_2, \ldots, l_K$. In what follows, we provide a mathematical characterization of the generalized survival signature and express its relation to system reliability.

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

$$\phi(l_1, l_2, \ldots, l_K) = \left[\prod_{k=1}^{K} \binom{n_k}{l_k}^{-1}\right] \times \sum_{\mathbf{x} \in S_l} \Psi(\mathbf{x}). \tag{5}$$

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

$$P\{T > \tau\} = \sum_{l_1=0}^{n_1} \cdots \sum_{l_K=0}^{n_K} \left[\phi(l_1, \ldots, l_K) \prod_{k=1}^{K} \left(\binom{n_k}{l_k} [F_k(\tau)]^{n_k - l_k} [1 - F_k(\tau)]^{l_k}\right)\right], \tag{6}$$

where $F_k(\tau)$ denotes the time-to-failure CDF for components of class $k \in \{1, 2, \ldots, K\}$. Computing Equation (6) 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. Kelkinnama and Eryilmaz (2023) considered coherent systems composed of different types of components that are monitored at one or two inspection times, and employed survival signatures to obtain dynamic reliability measures, e.g., mean residual live and mean inactive times. 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. For example, Reed et al. (2019) reported that the algorithm was unable to compute the two-terminal survival signature of a 12×12 grid system with 144 vertices, 264 edges, and two classes of components due to memory requirements (RAM). Xu et al. (2019) introduce an alternative method to compute the survival signature based on reliability block diagram and universal generating function. Their method computes the u-function of components connected in series, parallel, or bridge system through composition operators and then computes the survival signature from the reliability block diagram of the overall system. The method is limited due to large memory requirements and 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 (using 64 threads on an AMD Ryzen Threaddripper 3990X 64-Core Processor) to estimate the survival signature based on $1e4$ samples. Behrensdorf et al. (2021) report the limitations of the method for larger and more complex network and suggest applying advanced MC methods to reduce the number of samples needed. 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 research described above (in this paragraph), MC simulation has also been applied as a means of estimating system reliability given the survival signature (Patelli et al., 2017).

2.3 Other Closed Related Work

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 a minimum number 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). 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 $\phi(l_1, l_2)$ 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 $\phi(l_1, l_2)$ for $l_1 = 0, 1, \ldots, n_1$, and $l_2 = 0, 1, \ldots, n_2$. 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 (6) as

$$P\{T > \tau\} = \sum_{l_1=0}^{n_1} \sum_{l_2=0}^{n_2} \left[\phi(l_1, l_2) \prod_{k=1}^{2} \left(\binom{n_k}{l_k} [F_k(\tau)]^{n_k - l_k} [1 - F_k(\tau)]^{l_k} \right) \right]. \tag{7}$$

The main difficulty in this approach lies in obtaining the values of $\phi(l_1, l_2)$ for every combination of l_1 and l_2. We analyze four methods to estimate $\phi(l_1, l_2)$, $l_1 = 0, 1, \ldots, n_1$, $l_2 = 0, 1, \ldots, n_2$, 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 $l_1 = 0, 1, 2, \ldots, n_1$ and $l_2 = 0, 1, 2, \ldots, n_2$, 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 $\phi(l_1, l_2)$. For a given replication, let

$$\begin{aligned} q_i^1 &= k \quad \text{if } i \in \mathcal{N}_1 \text{ is the } k\text{th node to fail in } \mathcal{N}_1, \text{ and} \\ q_i^2 &= k \quad \text{if } i \in \mathcal{N}_2 \text{ is the } k\text{th node to fail in } \mathcal{N}_2. \end{aligned} \tag{8}$$

For $l_1 = 0, \ldots, n_1$ and $l_2 = 0, \ldots, n_2$, define $\mathbf{x}(l_1, l_2)$ such that components $i \in \mathcal{N}_1$ are UP if $q_i^1 > n_1 - l_1$ and DOWN otherwise, and similarly components $i \in \mathcal{N}_2$ are UP if $q_i^2 > n_2 - l_2$ and DOWN otherwise. Thus, the state vector $\mathbf{x}(l_1, l_2)$ 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 $j = 1, \ldots, \mathsf{M}$ yields a state vector $\mathbf{x}_j(l_1, l_2)$ for every pair (l_1, l_2). In replication j, we evaluate $\Psi(\mathbf{x}_j(l_1, l_2))$ for all $l_1 = 0, \ldots, n_1$ and $l_2 = 0, \ldots, n_2$, where $\Psi(\mathbf{x}_j(l_1, l_2)) = 1$ if there exists a path of functioning nodes from s to t in state $\mathbf{x}_j(l_1, l_2)$, and $\Psi(\mathbf{x}_j(l_1, l_2)) = 0$ otherwise. Hereafter, we refer to the collection $\Psi(\mathbf{x}_j(l_1, l_2))$, $l_1 = 0, 1, \ldots, n_1$; $l_2 = 0, 1, \ldots, n_2$, as the *system state matrix* for replication j. After completing M MC replications, we estimate $\phi(l_1, l_2)$ by

$$\phi(l_1, l_2) = \frac{\sum_{j=1}^{\mathsf{M}} \Psi(\mathbf{x}_j(l_1, l_2))}{\mathsf{M}}. \tag{9}$$

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

The Naive approach is summarized in Algorithm 1. In this approach, the value of $\Psi(\mathbf{x}_j(l_1, l_2))$ is evaluated for each $l_1 = 0, 1, \ldots, n_1$ and $l_2 = 0, 1, \ldots, n_2$ 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 $O(n_1 n_2 m \mathsf{M})$.

Algorithm 1: 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 (8)
    ▷ Turn DOWN $q_i^1$ for $i = 1, \ldots, n_1$, and $q_i^2$, for $i = 1, \ldots, n_2$
    for $l_1 = 0$ to $n_1$ do
        for $l_2 = 0$ to $n_2$ do
            ▷ Run a BFS from node $s$
            if the BFS reaches $t$ then
                $\phi(l_1, l_2) \leftarrow \phi(l_1, l_2) + 1$
            end
            ▷ Turn $q^2_{n_2 - l_2}$ UP
        end
        ▷ Turn $q^1_{n_1 - l_1}$ UP
    end
end
$\Phi \leftarrow \frac{1}{\mathsf{M}}\Phi$
return Φ
```

3.2 Incremental Search Approach

The incremental_search (IS) method exploits the fact that $\mathbf{x}_j(l_1, l_2)$ and $\mathbf{x}_j(l_1, l_2 + 1)$ are identical except for the addition of the node $q^2_{n_2 - l_2}$. Using this property, IS computes an entire row of the system state matrix (i.e., $\Psi(\mathbf{x}_j(l_1, -))$ 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 7–13 of Algorithm 1 compute the row $\Psi(\mathbf{x}_j(l_1, -))$ by running BFS n_2+1 times. The IS method is identical to Algorithm 1 with the exception that we do not re-initialize BFS for (l_1, l_2) with $l_2 > 1$; rather, we update the search from (l_1, l_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 a constant number of times. 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). The 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 1 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, l_2^*, for which $\Psi(\mathbf{x}_j(l_1, l_2)) = 0$. Once l_2^* is obtained, we have

$$\begin{aligned}&\Psi(\mathbf{x}_j(l_1,0)) = \Psi(\mathbf{x}_j(l_1,1)) = \cdots = \Psi(\mathbf{x}_j(l_1,l_2^*)) = 0, \text{ and}\\ &\Psi(\mathbf{x}_j(l_1,l_2^*+1)) = \Psi(\mathbf{x}_j(l_1,l_2+2)) = \cdots = \Psi(\mathbf{x}_j(l_1,n_2)) = 1,\end{aligned} \tag{10}$$

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

The problem of finding l_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 l_2^*. The single-objective optimization approach is based on solving the single-objective maximum capacity path problem once for each value of $l_1 = 0, 1, \dots, n_1$, 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 (8). Then, for fixed l_1, associate a weight u_i to each node $i \in \mathcal{N}$ according to

$$u_i = \begin{cases} q_i^2, & i \in \mathcal{N}_2, \\ 0, & \text{if } i \in \mathcal{N}_1 \text{ and } q_i^1 \le n_1 - l_1, \\ \infty, & \text{otherwise.} \end{cases} \tag{11}$$

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 l_2^* is then obtained by solving the maximum capacity path (MCP) problem

$$v^* = \max_{p \in \mathcal{P}} \{\min\{u_k : k \in p\}\}, \tag{12}$$

and setting $l_2^* = n_2 - v^*$. In Equation (12), $\min\{u_k : k \in p\}$ 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 MCP problem can be solved using a slight modification of Dijkstra's algorithm. In the case of the node-capacitated MCP problem of Equation (12), 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

$$\mathsf{d}(j) = \max\{\mathsf{d}(j), \min\{\mathsf{d}(i), u_j\}\}. \tag{13}$$

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 2 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 2 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 3 states the SO approach. Noting that the Dials-MaxCapPath algorithm provides l_2^* for each value of l_1, steps 8–12 of Algorithm 3 populate the corresponding row $\Psi(\mathbf{x}_j(l_1, -))$ according to Equation (10) by simply adding 1 to any entry for which $l_2 > l_2^*$. 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 3, 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.

For every value of l_1, the algorithm associates to each node $i \in \mathcal{N}$ a capacity u_i according to Equation (11); for $l_1 = 2$, the algorithm obtains the network in Figure 1(b). Then, the algorithm solves the MCP. At termination, Algorithm 2 provides a tree of maximum capacity paths rooted in s, and $v^* = 4$ is the capacity of the s-t path in this tree (blue in Figure 1(b)). Therefore, for any $l_2 > l_2^*$ $(= n_2 - v^* = 4 - 4 = 0)$,

Algorithm 2: Dials-MaxCapPath

```
1  bucket[$i$] $\leftarrow \varnothing,\ \forall i \in \{0, 1, \ldots, n\}$
2  d($i$) $\leftarrow 0,\ \forall\ i \in \mathcal{N} \setminus \{s\}$
3  d($s$) $\leftarrow n$
4  $idx \leftarrow n$
5  bucket[$idx$] $\leftarrow$ bucket[$idx$] $\cup \{s\}$
6  while $idx \geq 0$ do
7      if bucket[$idx$] $\neq \varnothing$ then
8          for $i \in$ bucket[$idx$] do
9              for $j \in \Gamma_i^+$ do
10                 if $d(j) < \min\{d(i), u_j\}$ then
11                     if $d(j) > 0$ then
12                         bucket[$d(j)$] $\leftarrow$ bucket[$d(j)$] $\setminus \{j\}$
13                     end
14                     $d(j) \leftarrow \min\{d(i), u_j\}$
15                     bucket[$d(j)$] $\leftarrow$ bucket[$d(j)$] $\cup \{j\}$
16                 end
17             end
18         end
19     end
20     $idx \leftarrow idx - 1$
21 end
22 return d($t$)
```

the network is UP. The resulting system state matrix is shown in Table 1, where the row corresponding to $l_1 = 2$ is highlighted in blue. Observe that the only value of l_2 for which the network is DOWN is 0, which corresponds to $l_2^* = 0$, and is formatted in red. □

Next, we evaluate the complexity of SO, starting by stating the complexity of Algorithm 2. 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 2 is $O(n + m)$. Since Algorithm 3 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 $n \leq m$, and hence the same time complexity of IS.

Algorithm 3: SO

```
1  $\Phi \leftarrow \mathbf{0}$
2  for $j = 1$ to M do
3      ▷ Simulate a permutation of $\mathcal{N}_1$ and a permutation of $\mathcal{N}_2$
4      for $l_1 = 0$ to $n_1$ do
5          ▷ Update $u_i,\ i \in \mathcal{N}$, according to Equation (11)
6          $v^* \leftarrow$ SO-MaxCapPath($\mathcal{G}$)
7          $l_2^* \leftarrow n_2 - v^*$
8          for $l_2 = 0$ to $n_2$ do
9              if $l_2 > l_2^*$ then
10                 $\phi(l_1, l_2) \leftarrow \phi(l_1, l_2) + 1$
11             end
12         end
13     end
14 end
15 $\Phi \leftarrow \Phi/$M.
16 return $\Phi$
```

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

	$\Psi(\mathbf{x}_j(l_1, l_2))$				
$l_1 \backslash l_2$	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

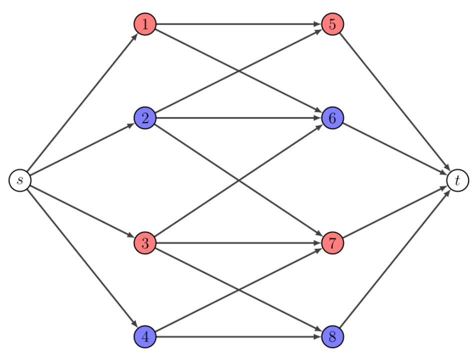
3.4 Bi-Objective Optimization Approach

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.

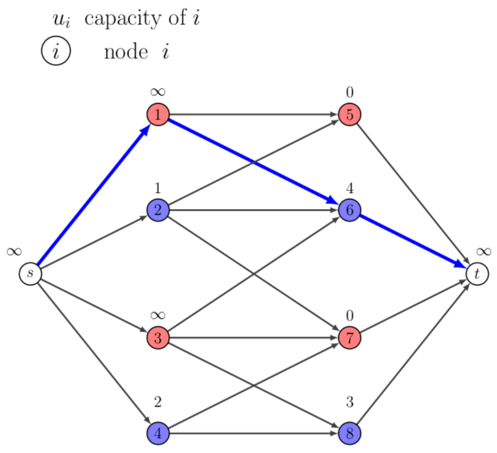
As with the previous approaches, we begin each replication by generating $q_i^1, i \in \mathcal{N}_1$, and $q_i^2, i \in \mathcal{N}_2$ according to Equation (8). To every node $i \in \mathcal{N}$, associate two weights according to

$$u_i^e = \begin{cases} q_i^e, & i \in \mathcal{N}_e, \\ \\ \infty, & i \in \mathcal{N}/\mathcal{N}_e, \end{cases} \qquad e = 1, 2. \tag{14}$$

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



(a) Heterogeneous s-t network.



(b) Network for $l_1 = 2$.

Figure 1: Computation of system state matrix row corresponding to $l_1 = 2$ using SO.

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 can be defined as

$$\max_{p \in \mathcal{P}} \{c^1(p), c^2(p)\}, \tag{15}$$

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 $\Psi(\mathbf{x}(l_1, l_2)) = 0$, for all $l_1 = 0, 1, \ldots, n_1$ and $l_2 = 0, 1, \ldots, n_2$. Letting (v_1^*, v_2^*) denote such a non-dominated point, the network is UP (i.e., $\Psi(\mathbf{x}(l_1, l_2)) = 1$) for every point (l_1, l_2) with $v_1^* > n_1 - l_1$ and $v_2^* > n_2 - l_2$. Furthermore, the existence of such a non-dominated point is guaranteed for any (l_1, l_2) in which $\Psi(\mathbf{x}(l_1, l_2)) = 1$. We record this result in the following theorem.

Theorem 3.1. $\Psi(\mathbf{x}(l_1, l_2)) = 1$ *if and only if there exists a non-dominated point* (v_1^*, v_2^*) *such that* $v_1^* > n_1 - l_1$ *and* $v_2^* > n_2 - l_2$.

Proof. $(\Rightarrow)$ Suppose $\Psi(\mathbf{x}(l_1, l_2)) = 1$. Then by definition of $\mathbf{x}(l_1, l_2)$, the system is UP when all components $i \in \mathcal{N}_1$ with $q_i^1 > n_1 - l_1$, and all components $i \in \mathcal{N}_2$ with $q_i^2 > n_2 - l_2$ 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 $c^1(p) > n_1 - l_1$ and $c^2(p) > n_2 - l_2$. 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, $v_1^* > n_1 - l_1$ and $v_2^* > n_2 - l_2$.
$(\Leftarrow)$ Conversely, suppose that there exists a non-dominated point (v_1^*, v_2^*) such that $v_1^* > n_1 - l_1$ and $v_2^* > n_2 - l_2$. Then, there exists an s-t path p with capacity $c^1(p) = v_1^*$ and $c^2(p) = v_2^*$, and hence $c^1(p) > n_1 - l_1$ and $c^2(p) > n_2 - l_2$. Thus, for all $i \in p$, $q_i^1 > n_1 - l_1$ and $q_i^2 > n_2 - l_2$, that is, all components $i \in p$ are UP with respect to the state $\mathbf{x}(l_1, l_2)$. Because $\mathbf{x}(l_1, l_2)$ contains an s-t path of functioning components, $\Psi(\mathbf{x}(l_1, l_2)) = 1$. □

We solve the BOMCP using the BO-MaxCapPath algorithm (given in Algorithm 4) 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 4, 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 2. 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 (13).

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 4 and BDijkstra are as follows: (i) Algorithm 4 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 4 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 4 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 4: BO-MaxCapPath

```
1  $\mathsf{N}_i \leftarrow 0$, $\mathsf{d}_i^1 \leftarrow 0$, $\mathsf{d}_i^2 \leftarrow 0$, $\mathsf{InH}[i] \leftarrow \mathsf{False}$, $i \in \mathcal{N} \setminus \{s\}$
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); $\mathsf{InH}[s] \leftarrow \mathsf{True}$
4  while $\mathsf{H} \neq \emptyset$ 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      $\mathsf{N}_{i^*} \leftarrow \mathsf{N}_{i^*} + 1$, $\mathsf{L}[i^*][\mathsf{N}_{i^*}] \leftarrow l^*$, $\mathsf{InH}[i^*] \leftarrow \mathsf{False}$
8      $l^{new} \leftarrow$ NewCandidateLabel($i^*$, $l^*$)
9      if $l^{new} \neq$ Null then
10         insert($l^{new}$, H), $\mathsf{InH}[i^*] \leftarrow \mathsf{True}$
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 $\mathsf{L}[t]$
```

Following the notation of Sedeño-Noda and Colebrook (2019), the data structure used in Algorithm 4 to 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 $\mathsf{L}[i][1]$, $\mathsf{L}[i][2], \ldots, \mathsf{L}[i][\mathsf{N}_i]$,

Algorithm 5: NewCandidateLabel(i^*, l^*)

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

Algorithm 6: RelaxationProcess(i^*, H, l^*)

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

stored in lexicographically decreasing order. Non-dominated points are represented by labels of the form

$$(i, \mathsf{d}^1, \mathsf{d}^2, j, r), \tag{16}$$

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 4 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: find-max(H), delete-max(H), insert(l, H), and increase-key(l, 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 5) and RelaxationProcess (Algorithm 6), 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 $v_1^* > n_1 - l_1$ and $v_2^* > n_2 - l_2$. Algorithm 7 states the BO approach.

Algorithm 7: BO

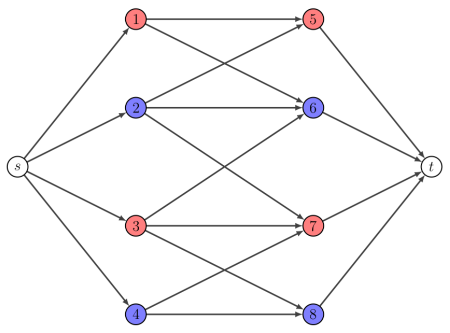
```
1  $\Phi \leftarrow \mathbf{0}$
2  for $j = 1$ to M do
3      ▷ Simulate a permutation of $\mathcal{N}_1$ and a permutation of $\mathcal{N}_2$
4      ▷ Update $u_i,\ i \in \mathcal{N}$, according to Equation (14)
5      $\Omega \leftarrow$ BO-MaxCapPath($\mathcal{G}$)                // $\Omega$ stores all $(v_1^*, v_2^*)$ points
6      for $(v_1^*, v_2^*) \in \Omega$ do
7          for $l_1 = 0$ to $n_1$ do
8              for $l_2 = 0$ to $n_2$ do
9                  if $v_1^* > n_1 - l_1$ and $v_2^* > n_2 - l_2$ then
10                     $\phi(l_1, l_2) \leftarrow \phi(l_1, l_2) + 1$
11                 end
12             end
13         end
14     end
15 end
16 $\Phi \leftarrow \Phi/$M.
17 return $\Phi$
```

We now establish the complexity of Algorithm 4. 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 find-max operation in

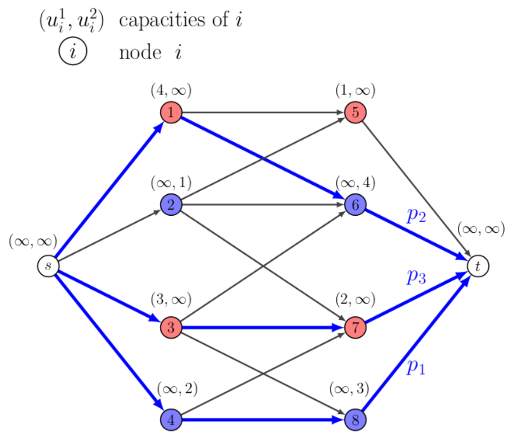
$O(1)$, a $\mathsf{delete\text{-}max}$ operation in $O(\log n)$, a series of assignment operations each having complexity $O(1)$, and, when the label returned by $\mathsf{NewCandidateLabel}$ is not Null, the algorithm performs an insert in $O(\log n)$ and another series of assignment operations in constant 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 4 is $O(nn_1 \log n + n_1^2 m + n_1 m \log n)$, which is $O(n_1^2 m + n_1 m \log n)$ provided that $n < m$. The overall complexity of BO is therefore $O(n_1^2 m\mathsf{M} + n_1 m \log n\mathsf{M})$. We illustrate BO with the following example.

Example 2. Consider the network in Figure 2(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 $P_1 = \{5, 7, 3, 1\}$ 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 (14) (see Figure 2(b)), 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(b)): $p_1 = s$ - 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 $\infty > 4 - l_1$ and $2 > 4 - l_2$, that is, the network is UP for $l_1 > -\infty$ ($l_1 \geq 0$) and $l_2 > 2$ (this



(a) Heterogeneous s-t network.



(b) Network after the updating of u_i.

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

	$\Psi(\mathbf{x}_j(l_1, l_2))$				
$l_1 \backslash l_2$	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

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. □

4 Computational Experiments

We implemented the four methods in section 3 and published the code 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 verified that (i) the estimated survival signature produced by all four methods were equivalent in every test instance; and (ii) the methods produced the (exact) survival signature for small-scale networks (such as the bridge system from Patelli et al. (2017); 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.

In what follows, we compare the run time of each approach using a variety of networks of different sizes

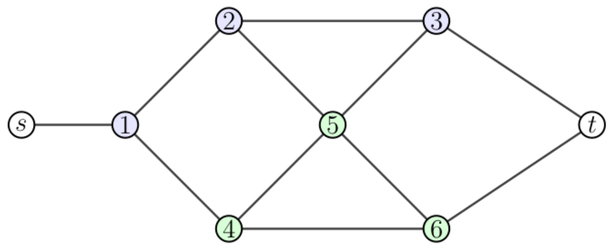
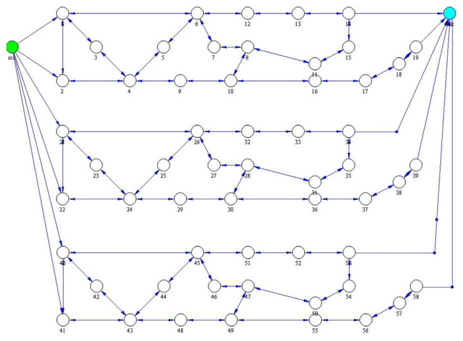


Figure 3: Bridge system from Patelli et al. (2017), where $\mathcal{N}_1 = \{1, 2, 3\}$ and $\mathcal{N}_2 = \{4, 5, 6\}$

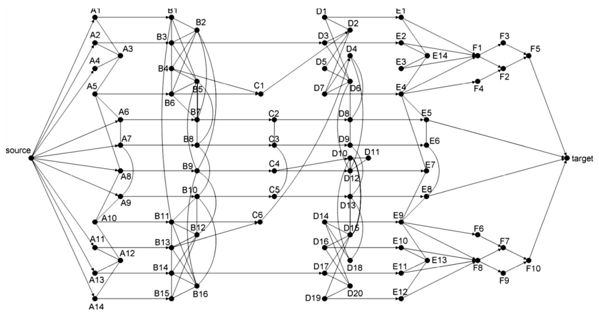
Table 3: Survival signature for the bridge system of Figure (3).

$l_1 \backslash l_2$	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

and with varying ratios of n_1/n_2. Initially, we evaluate how the algorithms perform with respect to a varying number of replications and verify that their running time is approximately linearly with M. For this round of experiments, we estimate the survival signature of two medium-sized networks, adapted from Sebastio et al. (2014), for $\mathsf{M} = 100, 500, 1000, 5000, 10000$ replications. The first network 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. We then record 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) shown in Table 4.



(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).

The second network used 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 divide the nodes in $\mathcal{N} \setminus \{s, t\}$ into $\mathcal{N}_1$ and $\mathcal{N}_2$ such that $n_1 = 40$ and $n_2 = 40$. As before, we record the time required by each algorithm to complete the survival signature estimation and divide this time by M to obtain the replication rates shown in Table 5.

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

All entries are given in seconds per replication					
	M = 100	M = 500	M = 1000	M = 5000	M = 10000
Naive	0.0088	0.0089	0.0082	0.0082	0.0080
IS	0.0006	0.0005	0.0006	0.0006	0.0005
SO	0.0006	0.0007	0.0007	0.0006	0.0006
BO	**0.0003**	**0.0003**	**0.0003**	**0.0003**	**0.0003**

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

All entries are given in seconds per replication					
	M = 100	M = 500	M = 1000	M = 5000	M = 10000
Naive	0.0360	0.0354	0.0353	0.0375	0.0352
IS	0.0017	0.0018	0.0018	0.0016	0.0017
SO	0.0017	0.0015	0.0015	0.0015	0.0015
BO	**0.0008**	**0.0008**	**0.0008**	**0.0007**	**0.0007**

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. Between IS and SO, neither seems to dominate the other, while Naive was at least one order of magnitude slower than the other algorithms.

Using the same networks, we next 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 round 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: 5/52, 12/45, 20/37, and 28/29. For the Airplane system, we consider the following combinations of n_1/n_2: 10/70, 20/60, 30/50, and 40/40. The results shown in Tables 6 and 7 reaffirm BO as the fastest method and suggest that BO is the most robust against variations in the ratio n_1/n_2.

To further investigate the robustness of the algorithms, we estimate the survival signature of a large-scale, generally structured network as described below. We generate a random geometric graph (RGG) with 350

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				
	$n_1 = 5,\ n_2 = 52$	$n_1 = 12,\ n_2 = 45$	$n_1 = 20,\ n_2 = 37$	$n_1 = 28,\ n_2 = 29$
Naive	0.0022	0.0042	0.0056	0.0082
IS	0.0001	0.0003	0.0004	0.0006
SO	0.0001	0.0003	0.0004	0.0006
BO	0.0001	**0.0002**	**0.0002**	**0.0003**

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				
	$n_1 = 10,\ n_2 = 70$	$n_1 = 20,\ n_2 = 60$	$n_1 = 30,\ n_2 = 50$	$n_1 = 40,\ n_2 = 40$
Naive	0.0125	0.0199	0.0243	0.0375
IS	0.0005	0.0008	0.0010	0.0016
SO	0.0004	0.0007	0.0009	0.0015
BO	**0.0003**	**0.0005**	**0.0005**	**0.0007**

nodes according to the following procedure: in the X-Y plane, we locate 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 generate coordinates x_i and y_i between 0 and 10 according to a uniform distribution, and we create and 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. For this example, we set $d = 1.5$ and the procedure generated $m = 7446$ arcs. 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 $\mathcal{N}_2$. We estimate the survival signature of this network for $\mathsf{M} = 5000$ replications and varying values of n_1/n_2. Figure (5) shows the resulting network for $n_1 = 149$ and $n_2 = 199$.

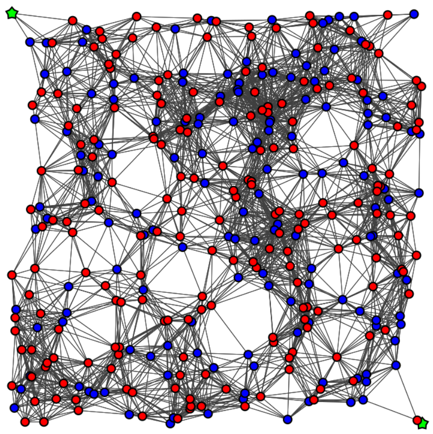


Figure 5: Network with $n = 350$, where $n_1 = 149$, $n_2 = 199$, and $m = 7446$ arcs.

Table 8 shows the results of the computational experiments performed with this RGG. For the 350-node RGG, the superiority of BO is highlighted in terms of both running time and robustness. In terms of run time rates, BO becomes orders of magnitude faster than the other approaches in most cases with only SO competing in the same order of magnitude but still around four times slower than BO. 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			
	$n_1 = 49,\ n_2 = 299$	$n_1 = 99,\ n_2 = 249$	$n_1 = 149,\ n_2 = 199$
Naive	3.8186	8.4882	9.6679
IS	0.1662	0.2526	0.3143
SO	0.0239	0.0431	0.0545
BO	**0.0078**	**0.0111**	**0.0121**

Using the same procedure, we generated two additional RGGs by varying the value of d to evaluate the performance of the algorithms with respect to generally structured networks with varying density. Notice that the density of the RGG discussed above (created with $d = 1.5$) is approximately 0.06. We set $d = 3.0$ and ran the procedure again; the resulting RGG has 25412 arcs and a density of 0.21. Lastly, we increased the value of d to 4.5 and created a third RGG, this one with 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.

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			
	$n_1 = 49,\ n_2 = 299$	$n_1 = 99,\ n_2 = 249$	$n_1 = 149,\ n_2 = 199$
Naive	6.3789	10.7940	13.0430
IS	1.6292	2.7340	3.3182
SO	0.0394	0.0704	0.0962
BO	**0.0116**	**0.0116**	**0.0121**

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			
	$n_1 = 49,\ n_2 = 299$	$n_1 = 99,\ n_2 = 249$	$n_1 = 149,\ n_2 = 199$
Naive	9.871	16.1800	21.1100
IS	7.5664	10.3460	12.1000
SO	0.0603	0.1088	0.1498
BO	**0.0177**	**0.0180**	**0.0185**

Tables 8, 9 and 10 reinforce the dominance of BO over the benchmarks and highlight its robustness against variations in n_1 and network density. In this round of experiments, SO is markedly faster than IS, whereas in the previous experiments this difference was not clear.

As our final objective in this section, we evaluate the performance of the algorithms in a realistic, large-scale network system. For this task, we estimate the survival signature of the 2000-bus power system, which has 4000 nodes and 29336 arcs and includes cycles and self-loops. This is a synthetic 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. Using BO, SO, and IS, we estimated the survival signature of the 2000-bus power system with $\mathsf{M} = 250$ replications for varying values of n_1/n_2. Because Naive required in excess of 10 hours per replication, we excluded it from this round of computational experiments. Table 11 shows the result for the other three algorithms.

Table 11: TAMU SmartGridCenter 2000-bus power system.

	All entries are given in seconds per replication		
	$n_1 = 999,\ n_2 = 2999$	$n_1 = 1499,\ n_2 = 2499$	$n_1 = 1999,\ n_2 = 1999$
IS	5.4018	5.7626	6.3502
SO	27.086	33.594	49.505
BO	**0.1863**	**0.1867**	**0.1983**

Table 11 confirms the results of previous experiments and highlights the advantages of BO over the other the benchmark methods. The BO algorithm scaled well and in the worst case ($n_1 = 1999, n_2 = 1999$) yielded a rate of 0.1983 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 6.35 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. Although in both cases the network considered was a large-scale system, the density in all three RGG cases 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, which is the case for many real systems. We did not pursue further investigation in this direction since BO decisively outperformed both SO and IS for every instance of considerable size.

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 multiobjective 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:

1. Generalizing our approach to estimate the two-terminal survival signature for networks with more than two component classes. Recent developments for multi-objective shortest path problems (de las Casas et al., 2021) may provide a natural extension to our bi-objective optimization-based MC approach.

2. Generalizing our approach to multi-state systems. Whereas we assume binary-state components and a binary-state system, the concept of signatures has been generalized to multi-state systems that present the possibility of non-binary performance levels (Eryilmaz and Tuncel, 2016; Qin and Coolen, 2022;

Yi et al., 2023). Extending the multi-objective optimization approach may enable multi-state survival signatures to be utilized for larger-scale systems.

3. Generalizing our approach to other reliability metrics. Although two-terminal reliability underlies many network reliability problems, there is a need to contemplate other network reliability metrics such as K-terminal reliability and coverage metrics. Extending our approach may provide an efficient way to estimate the survival signature for a broader variety of systems.

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, jun 2007. doi: 10.1016/j.ress.2006.04.021.

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.

F. Beichelt and P. Tittmann. *Reliability and Maintenance*. Chapman and Hall/CRC, may 2012. doi: 10.1201/b12095.

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

P. Caşcaval and S.-A. Floria. SDP algorithm for network reliability evaluation. In *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2017.

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, mar 2020. doi: 10.1109/tr.2019.2954894.

L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3): 410–421, August 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, November 1983.

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

F. J. Samaniego. On closure of the IFR class under formation of coherent systems. *IEEE Transactions on Reliability*, R-34:69 – 72, 05 1985. doi: 10.1109/TR.1985.5221935.

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, December 1991.

A. Sedeño-Noda and M. Colebrook. A biobjective Dijkastra algorithm. *European Journal of Operational Research*, 276, 2019. doi: 10.1016/j.ejor.2019.01.007.

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*, 2017.

P. Doulliez and E. Jamoulle. Transportation networks with random arc capacities. *RAIRO - Operations Research - Recherche Opérationnelle*, 6(V3):45–60, 1972.

T. Aven. Reliability evaluation of multistate systems with multistate components. *IEEE Transactions on Reliability*, R-34(5):473–479, dec 1985. doi: 10.1109/tr.1985.5222235.

C. Alexopoulos. A note on state-space decomposition methods for analyzing stochastic flow networks. *IEEE Transactions on Reliability*, 44(2):354–357, jun 1995. doi: 10.1109/24.387394.

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, nov 2018. doi: 10.1080/24725854.2017.1410598.

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. doi: 10.1109/tce.1958.6372698.

A. Satyanarayana and M. K. Chang. Network reliability and the factoring theorem. *Networks*, 13(1):107–120, 1983. doi: 10.1002/net.3230130107.

R. K. Wood. A factoring algorithm using polygon-to-chain reductions for computing k-terminal network reliability. *Networks*, 15(2):173–190, 1985. doi: 10.1002/net.3230150204.

R. K. Wood. Factoring algorithms for computing k-terminal network reliability. *IEEE Transactions on Reliability*, 35(3):269–278, aug 1986. doi: 10.1109/tr.1986.4335431.

J. M. Burgos and F. R. Amoza. Factorization of network reliability with perfect nodes i: Introduction and statements. *Discrete Applied Mathematics*, 198:82–90, jan 2016. doi: 10.1016/j.dam.2015.06.006.

H.-Y. Lin, S.-Y. Kuo, and F.-M. Yeh. Minimal cutset enumeration and network reliability evaluation by recursive merge and BDD. In *Proceedings of the Eighth IEEE Symposium on Computers and Communications. ISCC 2003*. IEEE Comput. Soc, 2003. doi: 10.1109/iscc.2003.1214299.

G. Hardy, C. Lucet, and N. Limnios. k-terminal network reliability measures with binary decision diagrams. *IEEE Transactions on Reliability*, 56(3):506–515, September 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, jun 2007. doi: 10.1109/tr.2007.896770.

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 and 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, March 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, sep 2021. doi: 10.1109/tr.2021.3049957.

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, June 1986.

J. E. Ramirez-Marquez and D. W. Coit. A monte-carlo simulation approach for approximating multi-state two-terminal reliability. *Reliability Engineering & System Safety*, 87(2):253–264, feb 2005. doi: 10.1016/j.ress.2004.05.002.

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, September 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, aug 2017. doi: 10.1016/j.ress.2017.01.021.

S. Reed, M. Löfstrand, and J. Andrews. An efficient algorithm for computing exact system and survival signatures of k-terminal network reliability. *Reliability Engineering and 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, dec 2021. doi: 10.1016/j.ress.2021.107935.

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, September 2011.

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

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.

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. *World Academy of Science, Engineering and Technology*, 3, 2007a.

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, 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.

M. Shaked and A. Suarez-Llorens. On the comparison of reliability experiments based on the convolution order. *Journal of the American Statistical Association*, 98(463):693–702, sep 2003. doi: 10.1198/016214503000000602.

J. Navarro and R. Rubio. Computations of signatures of coherent systems with five components. *Communications in Statistics - Simulation and Computation*, 39(1):68–84, dec 2009. doi: 10.1080/03610910903312185.

G. Da, B. Zheng, and T. Hu. On computing signatures of coherent systems. *Journal of Multivariate Analysis*, 103(1):142–150, jan 2012. doi: 10.1016/j.jmva.2011.06.015.

G. Da, P. S. Chan, and M. Xu. On the signature of complex system: A decomposed approach. *European Journal of Operational Research*, 265(3):1115–1123, mar 2018. doi: 10.1016/j.ejor.2017.08.052.

H. Yi and L. Cui. A new computation method for signature: Markov process method. *Naval Research Logistics (NRL)*, 65(5):410–426, aug 2018. doi: 10.1002/nav.21811.

I. Gertsbakh and Y. Shpungin. Combinatorial approaches to monte carlo estimation of network lifetime distribution. *Applied Stochastic Models in Business and Industry*, 20(1):49–57, jan 2004. doi: 10.1002/asmb.514.

Y. Shpungin. Combinatorial approach to reliability evaluation of network with unreliable nodes and unreliable edges. *International Journal of Electrical and Computer Engineering*, 1(12):4095–4099, 2007b.

I. Gertsbakh, Y. Shpungin, and F. Spizzichino. Signatures of coherent systems built with separate modules. *J. Appl. Prob.*, 48:843–855, 2011.

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. doi: 10.1002/asmb.2420.

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 and System Safety*, 2021.

M. Kelkinnama and S. Eryilmaz. Some reliability measures and maintenance policies for a coherent system composed of different types of components. *Metrika*, 86(1):57–82, mar 2023. doi: 10.1007/s00184-022-00862-5.

S. Eryilmaz and A. Tuncel. Generalizing the survival signature to unreparable homogeneous multi-state systems. *Naval Research Logistics*, 63, 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, apr 2022. doi: 10.1017/s0269964821000103.

S. Reed. An efficient algorithm for exact computation of system and survival signatures using binary decision diagrams. *Reliability Engineering and 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)*. IEEE, aug 2019. doi: 10.1109/qr2mse46217.2019.9021190.

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, mar 2023. doi: 10.1016/j.ress.2022.108982.

E. Patelli, G. Feng, F. P. A. Coolen, and T. Coolen-Maturi. Simulation methods for system reliability using survival signature. *Reliability Engineering and 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*, pages 1–14, 2021. doi: 10.1109/TR.2020.3047757.

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, jul 2017a. doi: 10.1109/tpwrs.2016.2616385.

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, aug 2017b. doi: 10.3390/en10081233.

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, nov 2018. doi: 10.1109/tpwrs.2018.2813525.

P. M. de las Casas, A. Sedeño-Noda, and R. Borndörfer. An improved multiobjective shortest path algorithm. *Computers & Operations Research*, 135:105424, nov 2021. doi: 10.1016/j.cor.2021.105424.

J. Qin and F. P. A. Coolen. Survival signature for reliability evaluation of a multi-state system with multi-state components. *Reliability Engineering & System Safety*, 218:108129, feb 2022. doi: 10.1016/j.ress.2021.108129.

H. Yi, N. Balakrishnan, and X. Li. Multi-state joint survival signature for multi-state systems with shared multi-state components. *Methodology and Computing in Applied Probability*, 25(1), mar 2023. doi: 10.1007/s11009-023-10023-4.