7-2021

Efficiently Estimating Survival Signature and Two-Terminal Reliability of Heterogeneous Networks through Multi-Objective Optimization

Daniel Bruno Lopes da Silva
University of Arkansas, Fayetteville

Efficiently Estimating Survival Signature and Two-Terminal Reliability of Heterogeneous Networks through Multi-objective Optimization

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Industrial Engineering

by

Daniel Bruno Lopes da Silva
Federal University of Alagoas, Brazil
Bachelor of Science in Industrial Engineering, 2017
Federal University of Pernambuco, Brazil
Master of Science in Industrial Engineering, 2019

July 2021
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Kelly M Sullivan, Ph.D.
Thesis Director

Burak Eksioglu, Ph.D.
Committee Member

Haitao Liao, Ph.D.
Committee Member

Abstract

The two-terminal reliability problem is a classical reliability problem with applications in wired and wireless communication networks, electronic circuit design, computer networks, and electrical power distribution, among other systems. However, the two-terminal reliability problem is among the hardest combinatorial problems and is intractable for large, complex networks. Several exact methods to solve the two-terminal reliability problem have been proposed since the 1960s, but they have exponential time complexity in general. Hence, practical studies involving large network-type systems resort to approximation methods to estimate the system's reliability. One attractive approach for quantifying the reliability of complex systems is to use signatures, but even signature-based approaches in computing exact network reliability may become computationally prohibitive as the number of components grows, and simulation-based approximations, such as Monte Carlo algorithms, are generally required. Nonetheless, the computation of the network's signature poses a major challenge in terms of computational time, especially when considering large, heterogeneous networks. Motivated by this, we propose a MC-survival signature based method to estimate two-terminal reliability for heterogeneous networks through multi-objective optimization. We formulate the problem of estimating the multi-dimensional survival signature of a network with heterogeneous components as a repeated multi-objective maximum capacity path problem and we present a fast and memory-efficient, Dijkstra-like algorithm to solve it. To the best of our knowledge, this is the first work to point out the relationship between the multi-dimensional survival signature computation and a multi-objective optimization problem. We empirically validate our method and perform computational experiments to compare its performance against two other approaches. The results of the experiments shows that our method is much faster than the other two approaches and can be used with a larger number of replications so to improve the accuracy of the reliability estimation.

Acknowledgements

Firstly, I would like to express my debt of gratitude and my appreciation to my advisor Dr. Sullivan for his guidance, patience, support, motivation and kindness during this journey. I truly enjoyed every aspect of working with him, and I was greatly benefited from having Dr. Sullivan always pointing out the best course of actions in every struggle I faced in my program. I gratefully acknowledge Dr. Eksioglu and Dr. Liao for serving in my committee and for their valuable feedback. I thank my UARK friends Hieu, Susan, Neel, Macy and Nick Boardman for our conversations, which made the journey more colorful. I thank my wife, Rafaela, for her continuing support and for her love. Finally, I would like to acknowledge the love and support from my parents Adelmo and Marcia, from my siblings Priscila, Paulo Wagner and José Henrique, and from my parents-in-law Valda and Jorge.

This material is based upon work supported by the National Science Foundation under Grant No. CMMI-1751191.

Dedication

To my wife, Rafaela, to my parents Adelmo and Marcia, and to my siblings Priscila, Paulo Wagner, and José Henrique.

Contents

1 Introduction **1**

2 Background and Literature Review **3**

2.1 Signatures 6

2.1.1 System Signature and D-Spectrum 6

2.1.2 Survival Signature 8

3 Methodology **11**

3.1 Naive Approach 13

3.2 Single-Objective Optimization Approach 15

3.3 Multi-Objective Optimization Approach 20

4 Computational Experiments **29**

4.1 Validation of Approaches 29

4.2 Comparison of Approaches 31

5 Considerations and Future Research **34**

List of Figures

1 SO survival signature computation 19

2 MO survival signature computation 28

3 Bridge system 29

4 Comparison of the methods 30

5 Network for computational experiments 32

6 Reliability curve 33

List of Tables

1 Structure function from SO example. 20

2 Structure function from MO example. 29

3 Survival signature of the bridge system 30

4 Estimated signature 31

5 Performance analysis of the Naive approach. 32

6 Performance analysis of the SO approach. 33

7 Performance analysis of the MO approach. 33

8 Complexity of Naive, SO, and MO approaches. 35

1 Introduction

Networks underpin every aspect of today's global economy and social organization from global vaccine supply chains to the Internet, from transportation and communication to protein-protein interactions, from underwater earthquake monitoring systems to autonomous vehicles, and from pandemic spread modeling to illegal supply chains (such as drugs and human trafficking). Network models constitute such a powerful analytical tool for analyzing interactions between components of a system and their reactions on the system itself that virtually every system can be naturally represented by a network. It is impossible, therefore, to overestimate the importance of creating mechanisms to understand complex networks, be it to sustain and protect or to disrupt and destroy them.

One can make a distinction when studying network reliability regarding the causes of network failure. Some failures are consciously caused by hostile agents who intend to destroy, disrupt, or take control over the network. This type of attack can be carried out by terrorist groups or adversary nations, but hacker attacks on virtual networks are of particular importance since they are becoming more frequent, more sophisticated, and potentially disastrous. The shutdown of major fuel pipelines, the take over of hospitals' data centers, and the disruption of meat plants this year, all due to cyberattacks, are just some of the latest examples.

In this paper, however, we focus on network failures caused by random events such as environmental conditions, wearout, or battery depletion. Network reliability in this sense is concerned with the probability that a system will be operating at a required level after some time in the presence of these random events. Such a general definition can be taken under different perspectives and opens up a wide variety of questions which in turn creates a numerous set of paths to address each of these questions, hence the massive literature on network reliability.

Network failure can be defined in different ways depending on the application. One of the most commonly used models, however, is the two-terminal model, which considers a

network operational as long as two pre-defined nodes can communicate with each other. Thus, in this model, a failure occurs when the two specified nodes are no longer connected. The two-terminal reliability problem is a classical reliability problem with applications in wired and wireless communication networks, electronic circuit design, computer networks, and electrical power distribution, among other systems. With the emergence of massive networked structures with thousands or even millions of components, efficient algorithms for computing or estimating network reliability remain an important research topic.

The two-terminal reliability problem, like many network reliability problems, has been proven to be intractable for large, complex networks. Several exact methods to assess two-terminal reliability have been proposed since the 1960s, and, in general, they are based on enumeration of minimal paths or cuts, sum of disjoint products, factorization, or binary decision diagrams. Hence, practical studies involving large network-type systems resort to approximation methods to estimate the system's reliability such as failure frequency approximation [1] and Monte Carlo simulation [2, 3, 4].

One attractive approach for quantifying the reliability of complex systems is to use signatures [5, 6, 7]. The signature of a system is a function of only its design and its definition of failure. It thus allows for a separation of the structure's impact from the random components failure's impact on the system's reliability [8]. Nevertheless, even signature-based approaches in computing exact network reliability may become computationally prohibitive as the number of components grows, and simulation-based approximations, such as Monte Carlo (MC) algorithms, are generally required.

Still, the combined use of MC with signatures is 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 whereas crude MC may have unbounded relative error [9]. Nonetheless, the computation of the network's signature poses a major challenge in terms of complexity/computational time, especially

when considering large, heterogeneous networks. With this motivation, Boardman and Sullivan [10] presented a combined MC/signature approach to estimate the coverage reliability of a wireless sensor network — albeit one with homogeneous components — in which a uni-dimensional signature can be estimated by a MC algorithm that amount to solving a maximum capacity path/route problem (see [11, 12, 13]) at each iteration. To solve this problem, they proposed an algorithm based on Dial's implementation of Dijkstra's algorithm.

Building on the work of Boardman and Sullivan [10], we extend this results to the case of a two-terminal system with heterogeneous components, which require the use of a multi-dimensional survival signature. To the best of our knowledge, this is the first work to point out the relationship between the multi-dimensional survival signature computation and multi-objective optimization problem. We formulate the problem of estimating the signature of a network with heterogeneous components as a repeated multi-objective maximum capacity path problem and we present a fast and memory-efficient, Dijkstra-like algorithm to solve it based on an extension of the work of [14]. Therefore, the main contribution of this paper is twofold: (1) the relationship between the computation of multi-dimensional survival signature and a multi-objective optimization problem; (2) a fast and memory-efficient algorithm to estimate the two-terminal reliability of heterogeneous systems.

The remaining of this paper is organized as follows. Section 2 provides background material and summarizes the literature related to network reliability with a focus on two-terminal reliability problems as well as signatures. In Section 3 we introduce three approaches to deal with the problem of estimating two-terminal reliability for heterogeneous networks. In Section 4, we conduct computational experiments to validate and demonstrate our algorithms of Section 3. Finally, Section 5 presents our conclusions and directions for future research.

2 Background and Literature Review

In a network where some elements fail independently of the other components according to a known probability, the two-terminal reliability problem is to determine the probability

that there is at least one functional path between two specified terminal nodes s and t. The two-terminal reliability problem is a special case of the K-terminal reliability problem, which is to determine the probability that there is a functional path between every pair of the K terminal nodes. The two-terminal and the K-terminal reliability problems have been shown to be #P-Complete, and therefore at least as hard as any NP-Complete problem [15, 16, 17].

Many exact approaches to solve the two-terminal reliability problem have been proposed since the late 1950s. We refer the reader to [18] and [19], for a summary of early research in this area and to [20] 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. Both exact and approximation algorithms are usually based on one or more of the following methods: (1) sum of disjoint products (SDP), (2) cut and path-based state enumerations, (3) factoring algorithm, and (4) binary decision diagram (BDD). Classes (1) and (2) are closely related since SDP algorithms requires enumeration of minimal cuts in advance, and their major drawback is that their complexity is usually exponential [21]. Factoring algorithms have generally exponential time complexity as well [22], while BDD-based algorithms require large memory capacity and some pre-processing, which can present limitations when solving large, dense networks [23]. Other approaches to solve two-terminal reliability problems include neural networks [24, 25], decomposition algorithms [26], the cross-entropy method [27], evolutionary optimization techniques [28], failure frequency approximation [1], and various Monte Carlo simulation based methods [2, 3, 4].

Among the SDP methods, Jane and Yuan [29] proposed an algorithm for the generalized capacitated version of the two-terminal reliability problem, Datta and Goyal [30] proposed an exact SDP technique based on the Inclusion-Exclusion Principle, and Caşcaval and Floria [31] presented a method that combines SDP with multiple variables inversion to transform a structure function into a sum of disjoint products. Minimal cut sets were used by [32] and [22]. Gebre and Ramirez-Marquez [32] proposed an algorithm based on set-replacement and

element inheritance to obtain a network's minimal cut set while Silva *et al.* [22] presented a bounding algorithm based on ordered subset of minimal cuts and minimal paths to compute the all-terminal reliability upper and lower bounds. The multi-state two-terminal reliability at specified demand level problem was investigated in [33, 34]. Ramirez-Marquez *et al.* [33] proposed an algorithm to obtain the multi-state equivalent of binary path sets, while Zhang and Shao [34] presented a diameter-constrained approximation algorithm.

The use of BDD is a more recent approach and is efficient at manipulating Boolean expressions. Sebastio *et al.* [35] proposed a BDD bounding algorithm for the two-terminal reliability problem. Their algorithm is based on a novel search strategy to find important minimal paths and minimal cuts. Nevertheless, BDD-based algorithms have their specific drawbacks. Many of the BDD methods still rely on finding minimal cut/path sets and hence can have exponential time complexity. The memory requirement is another major disadvantage of BDD algorithms. In fact, the study conducted by Lê *et al.* [21] is focused on the memory usage when solving the two-terminal reliability problem through a BDD-method. They propose a strategy to minimize the impact of memory usage without significantly increasing the computational time.

More recent studies have applied signature-based approaches for computing or approximating reliability problems and specifically two-terminal reliability problems. An interesting example of this trend is the algorithm present in [23] and expanded in [36]. In [23], Reed proposes an exact dynamic programming algorithm to compute a system's signature based on a reduced ordered binary decision diagram (ROBDD) representation of the system. The algorithm demonstrated empirical gains compared to enumerative algorithms. Nonetheless, the BDD representation may be exponential in size [37, 38]. Reed *et al.* [36] address the K-terminal reliability problem by extending from the algorithm presented in [23] and combining it with BDD classical methods such as boundary set partition and factorization. Although efficient for most of the benchmark examples, this algorithm is constrained by significant memory usage in some cases (particularly in networks with a large number of edges) and it

applies only to undirected networks with unreliable edges and completely reliable nodes.

2.1 Signatures

In this section we introduce the concept of signatures, a technique that has been widely used for analyzing and comparing the reliability of multi-component system. To formally introduce these concepts, 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

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

Although structure functions provide one characterization of a system, they are difficult to compute and to compare [39]. These difficulties motivated seeking more practical forms to describe the system's dependence on its components. A very useful approach has been the use of signatures, the first of which was the system signature.

2.1.1 System Signature and D-Spectrum

The notion of a system signature was introduced by Samaniego [5] for coherent systems — i.e., in which all components can affect the system state and the structure function $\Psi(\mathbf{x})$ is non-decreasing in the component state vector $\mathbf{x}$ — composed of binary elements with independently and identically distributed (i.i.d.) failure times. System signature is also referred to as destruction spectrum (D-spectrum) [6, 8, 40] since these two characterizations are equivalent. Under the assumption of i.i.d. component lifetimes, the signature $\mathbf{s}$ of a coherent system of order n is an n-dimensional vector whose ith element is the probability

$$s_i = P\{T = W_{(i)}\}, \tag{2}$$

where T is the system lifetime and $W_{(1)}, W_{(2)}, \ldots, W_{(n)}$ are the order statistics of n i.i.d. component lifetimes. In other words, s_i represents the probability that the ith component failure causes the system to fail. Note that the signature vector $\mathbf{s}$ does not depend on the common lifetime distribution of the components and is, therefore, a pure measure of the system's design [5, 39, 41].

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 which ordinal failure number cause the system to fail. Then, s_i is the proportion of permutations in which the ith failure causes the system's failure. We now formalize this alternative definition of the system signature according to [42].

Definition 2.1. Let $1, 2, \ldots, n$, be the components of the system subject to failure. Denote by $\pi = (1, 2, \ldots, n)$ a permutation of system components. Suppose that initially they all are up. Then, begin to turn them down by moving along π from left to right. Fix the first component i_r when the system state becomes DOWN. The ordinal number r of this component in the permutation is called *anchor* of π and denoted by $r(\pi)$.

Because the $n!$ permutations are equally likely, the system signature is given by

$$s_i = \frac{\text{number of permutations with } r(\pi) = i}{n!}, \ i = 1, 2, \ldots, n. \tag{3}$$

Once the system signature is computed, the reliability of the system can be calculated by

$$P\{T > t\} = \sum_{i=1}^{n} s_i \overline{F}_{(i)}(t) = \sum_{i=1}^{n} s_i \sum_{j=0}^{i-1} \binom{n}{j} [F(t)]^j [\overline{F}(t)]^{n-j}, \tag{4}$$

where T is the system lifetime, $F(t)$ is the common lifetime CDF of components $i, \ldots, n$ and $\overline{F}(t) = 1 - F(t)$ is the common reliability function [42]. System signature and D-spectrum have been extensively used in applications such as comparison of coherent systems [39, 42, 43], lifetime estimation [44], analysis of queueing systems [45], and reliability [46, 47, 48, 49, 50].

For the case of multi-state systems, Gertsbakh, Shpungin and Spizzichino [51] explain that for systems with ν states ($\nu > 2$) in n binary components, the signature is a joint probability distribution for $(\nu-1)$ ordered, $\{1, \ldots, n\}$-valued random variables and therefore $(\nu-1)$ can be interpreted as a dimension for the signature. Following this understanding, the D-spectrum is sometimes defined as a multi-dimensional signature. In our work, we use the term multi-dimensional signature to refer to the signature of a system with heterogeneous components, that is, a system with multiple classes of components. To avoid ambiguity, we use the term multi-state signature for a system with more than two possible states, and the term multi-dimensional signature for a system with more than one type of components.

2.1.2 Survival Signature

The survival signature was introduced by Coolen and Coolen-Maturi [7] as an extension of the notion of system signature to the case of independent but not identically distributed components' lifetime, but preserving the main idea of separating the contribution of the system's structure from the components' failure probability to the system reliability [42]. Hence, for the case of i.i.d. components' lifetime, the survival signature is closely related to the system's signature. Let us first define the survival signature for the i.i.d. case.

Let $\phi(l)$, for $l = 0, \ldots, n$, denote the probability that the system functions if exactly l of its components function, and let the vector whose entries are the $\phi(l)$, for $l = 0, \ldots, n$, be denoted by Φ. Let S_l denote the set of state vectors in which $x_i = l$ for exactly l components, and observe that $|S_l| = \binom{n}{l}$. Given the component's i.i.d. failure time, all vectors in S_l are equally likely, and therefore

$$\phi(l) = \binom{n}{l}^{-1} \sum_{\mathbf{x} \in S_l} \Psi(\mathbf{x}),\ l = 0, 1, \ldots, n, \tag{5}$$

as shown by Coolen and Coolen-Maturi [7].

It is straightforward to show (see [7]) that survival signature and system signature satisfy the relationship

$$\phi(l) = \sum_{j=n-l+1}^{n} s_j, \tag{6}$$

The right-hand side of Equation (6) denotes the probability that at least $(n - l + 1)$ component failures are required for the system to fail, which is equal to probability that the system functions when there are exactly l components functioning, i.e., the left-hand side of Equation (6).

Although the survival signature can be applied to systems with i.i.d. components, the fundamental contribution of the survival signature is the generalization of the theory of signatures to system with multiple types of components. Following [7], consider a system with $K \geq 2$ types of components, where components of the same type have i.i.d. failure times and failure times of components of different types are independent but not identically distributed. Let n_k denote the number of components of type 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 type 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 type-k components function for each $k \in \{1, 2, \ldots, K\}$. For $K > 1$, let Φ denote the K-dimensional survival signature, which is a multi-dimensional matrix whose entries are $\phi(l_1, l_2, \ldots, l_K)$ for all the values of $l_1, l_2, \ldots, l_K$. Note that the multi-dimensional survival signature is a representation of a system with heterogeneous components and should not be confused with the multi-state D-spectrum or multi-state signature discussed in the previous subsection.

For $k \in \{1, 2, \ldots, K\}$, let $S_l^k \subseteq \{0,1\}^{n_k}$ denote the set of type-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 type k have i.i.d. failure times all state vectors $\mathbf{x}^k \in S_l^k$ are equally likely and therefore

$$\phi(l_1, l_2, \ldots, l_K) = \left[\prod_{k=1}^{K} \binom{n_k}{l_k}^{-1}\right] \times \sum_{\mathbf{x} \in S_l} \Psi(\mathbf{x}). \tag{7}$$

Given the survival signature Φ, Coolen and Coolen-Maturi [7] showed that

$$P\{T > t\} = \sum_{l_1=0}^{n_1} \cdots \sum_{l_K=0}^{n_K} \left[\phi(l_1, \ldots, l_K) \prod_{k=1}^{K} \left(\binom{n_k}{l_k} [F_k(t)]^{n_k - l_k} [1 - F_k(t)]^{l_k}\right)\right], \tag{8}$$

where $F_k(t)$ denotes the time-to-failure CDF for component of type $k \in \{1, 2, \ldots, k\}$. Computing Equation (8) is challenging because it requires evaluating all $\prod_{k=1}^{K}(n_k + 1)$ elements of Φ.

Some of the most important developments of the theory of survival signature are summarized next. In [52], the authors presented general results for coherent systems with multiple types of dependent components. In [53], Coolen-Maturi *et al.* introduced a joint survival signature for multiple systems with multiple types of components and with shared components between systems. In [54], the authors used the survival signature in nonparametric predictive inference for system reliability. Aslett *et al.* [55] applied the survival signature with Bayesian inference for system reliability. Eryilmaz and Tuncel [56] generalized the survival signature to unrepairable homogeneous multi-state systems with multi-state components, while in [57], the survival signature is used to study system reliability when failed components can be replaced by functioning components of the same type already in the system.

Still, the computation of the survival signature as well as the computation of the reliability for systems of realistic size poses a major challenge in terms of run-time and even in terms

of memory/storage and hence, in many problems, simulation methods are preferred. In [58], for instance, the authors present three simulation based algorithms for reliability estimation using survival signature.

3 Methodology

Although many other variants have been considered, the first K-terminal reliability problems considered undirected binary networks with nodes completely reliable and binary-state edges. By contrast, we define our problem over a directed network with perfectly reliable and binary-state nodes. In general, these assumptions are without loss of generality as undirected edges and/or unreliable edges can be accommodated by standard network transformations.

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 $i \in \mathcal{N}$. Nodes fail according to a specified probability distribution but arcs are completely reliable and therefore cannot fail. However, when a node fails all arcs that have this node as an extreme point are extinguished. The network has two specified nodes: the source s and the sink or terminal t. These nodes 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 nodes of the network can be classified according to their failure distribution. There are n_e nodes that fail independently according to a failure CDF, $F_e(t)$; we denote this subset of nodes by $\mathcal{N}_e$. Nodes in different groups are fully independently regarding failure. Besides s and t, the network can also have a subset of nodes that are completely reliable. To simplify the presentation of our methods, we consider networks with only two groups of nodes subject to failure, $\mathcal{N}_1$ and $\mathcal{N}_2$, and we assume without loss of generality that $n_1 \leq n_2$.

For convenience, we recall the definition of the multi-dimensional survival signature for the case in hand. Let $\phi(l_1, l_2)$ denote the probability that the network is functioning given

that exactly l_1 nodes from $\mathcal{N}_1$ and l_2 nodes from $\mathcal{N}_2$ are functioning. For $|\mathcal{N}_1| = n_1$ and $|\mathcal{N}_2| = n_2$, we denote the survival signature of the network, Φ, as 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 none of its nodes are functioning, and that $\phi(n_1, n_2) = 1$, that is, the network is functioning when all of its nodes are functioning. Additionally, we observe that this system cannot be deteriorated when the number of functioning nodes increases, which implies that the network thus defined is a coherent system.

Note that, provided Φ, $F_1(t)$, and $F_2(t)$, one can estimate the reliability of the network at any time t through Equation (8), which specializes to

$$P\{T > t\} = \sum_{l_1=0}^{n_1}\sum_{l_2=0}^{n_2}\left[\phi(l_1, l_2)\prod_{k=1}^{2}\left(\binom{n_k}{l_k}[F_k(t)]^{n_k-l_k}[1-F_k(t)]^{l_k}\right)\right]. \tag{9}$$

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 three different approaches to this problem: a naive approach, a single-objective optimization method, and a multi-objective optimization method. These three approaches follow the same framework to estimate the two-terminal reliability, differing only on the method to estimate the survival signature. The common framework used by the three approaches is presented below. We begin our discussion by the simplest and more intuitive method, the naive approach.

Algorithm 1: ReliabilityEstimation

1 **Initialization and network generation**
2 ▷ Initialize variables and generate network
3 **Survival signature estimation**
4 ▷ Estimate survival signature according to one of the three approaches
5 **Reliability estimation**
6 ▷ Estimate the reliability of the system through Equation (9)

3.1 Naive Approach

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

In each replication of the Naive approach, we independently simulate 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 $l_1 = 0, 1, 2, \ldots, n_1$ and $l_2 = 0, 1, 2, \ldots, n_2$ and assess the structure function for every one of the state vectors formed. For every combination of l_1 and l_2, we count the number of state vectors for which the network is up and divide this number by the number of replications simulated, which gives the estimate of $\phi(l_1, l_2)$. The Naive approach is formally stated below.

First, independently simulate a permutation of $\mathcal{N}_1$ and a permutation of $\mathcal{N}_2$ M times. Let

$$\begin{aligned} q_i^1 &= k \quad \text{if } i \in \mathcal{N}_1 \text{ is the } k\text{th node to fail in } \mathcal{N}_1, \text{ and} \\ q_i^2 &= k \quad \text{if } i \in \mathcal{N}_2 \text{ is the } k\text{th node to fail in } \mathcal{N}_2. \end{aligned} \tag{10}$$

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 are down otherwise, and similarly components $i \in \mathcal{N}_2$ are up if $q_i^2 > n_2 - l_2$ and are 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.

Next, evaluate $\Psi(\mathbf{x}(l_1, l_2))$ for all $l_1 = 0, \ldots, n_1$ and $l_2 = 0, \ldots, n_2$. Recall that our definition of failure is based on s-t connectivity. In the Naive approach, we assess the structure function for every state vector by running a breadth-first search (BFS) from s and verifying if t can be reached. Then,

$$\Psi(\mathbf{x}(l_1, l_2)) = \begin{cases} 1 & \text{if the BFS reaches } t, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

For every replication $j = 1, \ldots, \mathsf{M}$, we will obtain a state vector for every combination of l_1 and l_2. Let $\mathbf{x}_j(l_1, l_2)$ denote the state vector for replication j. Then, we can estimate $\phi(l_1, l_2)$ by

$$\phi(l_1, l_2) = \frac{\sum_{j=1}^{\mathsf{M}} \Psi(\mathbf{x}_j(l_1, l_2))}{\mathsf{M}}. \tag{12}$$

The Naive approach is summarized in Algorithm 2. This approach requires a BFS run $\mathsf{M} \times (n_1 + 1) \times (n_2 + 1)$ times with each BFS having a $O(m)$ complexity, where m is the number of edges in the network. The overall complexity is therefore $O(n_1 n_2 m \mathsf{M})$.

Algorithm 2: Naive-SurvSig

```
Φ ← 0
for j = 1 to M do
    ▷ Simulate a permutation of 𝒩₁ and a permutation of 𝒩₂
    ▷ Represent the permutations according to Equation (10)
    for l₁ = 0 to n₁ do
        for l₂ = 0 to n₂ do
            ▷ turn down components q_i^1 for i = 1, ..., n₁ − l₁,
            and components q_i^2, for i = 1, ..., n₂ − l₂
            ▷ Run a BFS from node s
            if the BFS reaches t then
                φ(l₁, l₂) ← φ(l₁, l₂) + 1
            end
        end
        φ(l₁, l₂) ← φ(l₁, l₂)/M
    end
end
return Φ
```

It is worth mentioning that there are potential improvements that could be used with the Naive method. For example, we could replace the loop over l_2 with a bisection search noting that $\Psi(\mathbf{x}_j(l_1, l_2))$ must be nondecreasing in l_2 when l_1 is fixed (because Ψ is nondecreasing in $\mathbf{x}$). The resulting approach, which we refer to as Naive-bisection would improve upon the complexity of the Naive by changing the n_2 to $\log n$. We did not pursue this improvements

however because we believe the methods based on optimization to be presented next are superior with respect to the Naive approach, and therefore the gains that could be obtained with the Naive approach would be overshadowed by the other approaches anyway.

3.2 Single-Objective Optimization Approach

As we have noted in the previous discussion, the bottleneck operation of the Naive approach is the use of breadth-first search to assess the state of the network at every changing of a component's state. In the single-objective optimization (SO) method, we extend the work of Boardman and Sullivan [10] and avoid the use of repeated BFS by solving an optimization problem. We explain the main ideas underlying the approach in the following lines.

Consider the Algorithm Naive-SurvSig again, and suppose we fix the value of l_1. Then, the loop on l_2 could alternatively attempt to identify the maximum value of l_2, l_2^*, for which $\Psi(\mathbf{x}(l_1, l_2)) = 0$. Once l_2^* is obtained, we would have that

$$\begin{aligned} &\Psi(\mathbf{x}(l_1, 0)) = \Psi(\mathbf{x}(l_1, 1)) = \cdots = \Psi(\mathbf{x}(l_1, l_2^*)) = 0, \text{ and} \\ &\Psi(\mathbf{x}(l_1, l_2^* + 1)) = \Psi(\mathbf{x}(l_1, l_2 + 2)) = \cdots = \Psi(\mathbf{x}(l_1, n_2)) = 1, \end{aligned} \tag{13}$$

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 discussed by [11, 12]. Boardman and Sullivan [10] solved a similar maximum capacity path problem as a subroutine in evaluating the D-spectrum of a network with homogeneous 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 $l_1 = 0, 1, \ldots, n_1$, with the additional consideration that the l_1 components from $\mathcal{N}_1$ that are up are uncapacitated, which we represent by assigning "∞" as their capacity.

To formalize the single-objective optimization approach, independently simulate a permutation of $\mathcal{N}_1$ and a permutation of $\mathcal{N}_2$, and record these permutations according to Equa-

tion (10). Then, for fixed l_1, associate a weight u_i to each node $i \in \mathcal{N}$ according to

$$u_i = \begin{cases} q_i^2, & i \in \mathcal{N}_2, \\ 0, & \text{if } i \in \mathcal{N}_1 \text{ and } q_i^1 \leq n_1 - l_1, \\ \infty, & \text{otherwise.} \end{cases} \tag{14}$$

In our implementation, we substitute ∞ by a number larger than $\max\{q_i^1 : i \in \mathcal{N}_1\}$ and $\max\{q_i^2 : i \in \mathcal{N}_2\}$, 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 node t. The value l_2^* is then obtained by solving the maximum capacity path problem

$$v^* = \max_{p \in \mathcal{P}} \{\min\{u_k : k \in p\}\}, \tag{15}$$

and setting $l_2^* = n_2 - v^*$. In Equation (15), $\min\{u_k : k \in p\}$ selects the smallest node capacity in the s-t path p since a path capacity is determined by its smallest node capacity, and then $\max_{p \in \mathcal{P}}$ selects the largest capacity among all s-t paths. Therefore, v^* represents the capacity of the last s-t path to fail, or more specifically, v^* represents the number of node failures that would cause the system to fail.

Pollack [11] observed that any shortest path algorithm, such as Dijkstra's, could be used to solve the maximum capacity path problem with the following slight modification: for the maximum capacity path problem with arc capacities $w_{i,j}, (i,j) \in \mathcal{A}$, the label of a node is initialized as $\mathsf{d}(i) = 0$ for all $i \in \mathcal{N} \setminus \{s\}$, and $\mathsf{d}(s) = \infty$, and the labels are updated (when considering an arc (i,j) leaving a node i whose label has been made permanent) according to

$$\mathsf{d}(j) = \max\{\mathsf{d}(j), \min\{\mathsf{d}(i), w_{i,j}\}\}. \tag{16}$$

Additionally, a node label with maximum value is made permanent in each iteration instead of minimum value, as in the case of shortest path. The case with node capacities u_i, $i \in \mathcal{N}$,

can be transformed to the arc-capacitated case by setting $w_{i,j} = \min\{u_i, u_j\}$, $\forall (i,j) \in \mathcal{A}$, resulting in the update in Equation (16).

In this work we use the heap version of Dijkstra's algorithm to solve the maximum capacity path problem in Equation (15). Algorithm 3 is an adaptation of the Heap-Dijkstra algorithm presented in [59]. The binary heap, H, used in Algorithm 3 allows us to efficiently manipulate the nodes $i \in \mathcal{N}$ with their associated labels. Specifically, H performs the following operations: (1) find and return a node of maximum label; (2) insert a new node i with its label; (3) increase the label of a node i from its current value to *value*; and (4) delete a node with maximum label. Those operations are denoted by find-max(H), insert(i, H), increase-key(*value*, i, H), and delete-max(H), respectively.

Algorithm 3: SO-MaxCapPath($\mathcal{G}$)

```
1  d(j) := 0   ∀ j ∈ N \ {s}
2  d(s) := ∞, pred(s) := 0
3  insert(s, H)
4  while H ≠ ∅ do
5      i := find-max(H); delete-max(H)          // Γ_i^+ is the set of successors of node i
6      for j ∈ Γ_i^+ do
7          value := min{d(i), u_j}
8          if d(j) < value then
9              if d(j) = 0 then
10                 d(j) := value, pred(j) := i
11                 insert(j, H)
12             end
13             else
14                 d(j) := value, pred(j) := i
15                 increase − key(value, j, H)
16             end
17         end
18     end
19 end
20 return d(t)
```

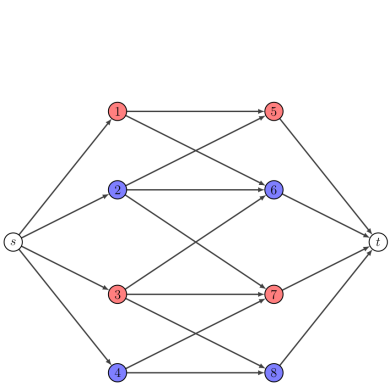
Algorithm 4: SO-SurvSig

$\Phi \leftarrow \mathbf{0}$
for $j = 1$ **to** M **do**
  ▷ Simulate a permutation of $\mathcal{N}_1$ and a permutation of $\mathcal{N}_2$
  for $l_1 = 0$ **to** n_1 **do**
    ▷ Update $u_i,\ i \in \mathcal{N}$, according to Equation (14)
    $v^* =$ SO-MaxCapPath$(\mathcal{G})$
    $l_2^* = n_2 - v^*$
    for $l_2 = 0$ **to** n_2 **do**
      if $l_2 > l_2^*$ **then**
        $\phi(l_1, l_2) \leftarrow \phi(l_1, l_2) + 1$
      end
    end
  end
end
$\Phi = \Phi/\mathsf{M}$.
return Φ

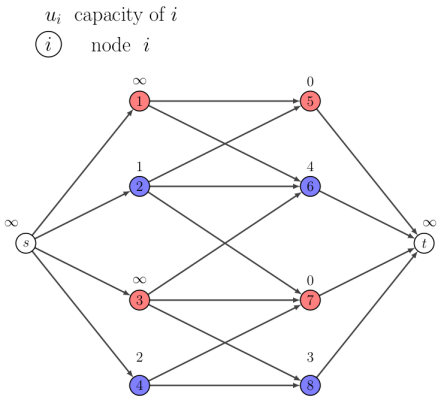
Algorithm 4 states the SO approach. Noting that the SO-MaxCapPath algorithm provides l_2^* for each value of l_1, we can populate the corresponding row of the survival signature matrix, $\phi(l_1, -)$ according to Equation (13) by simply adding 1 to any entry for which $l_2 > l_2^*$.

We demonstrate the SO approach with the following example. Consider the network in Figure 1(a). For this network, $n = 10$ nodes, $\mathcal{N}_1 = \{1, 3, 5, 7\}$ is represented in red, $\mathcal{N}_2 = \{2, 4, 6, 8\}$ is represented in blue. Suppose that in jth replication of the SO-SurvSig algorithm, 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.

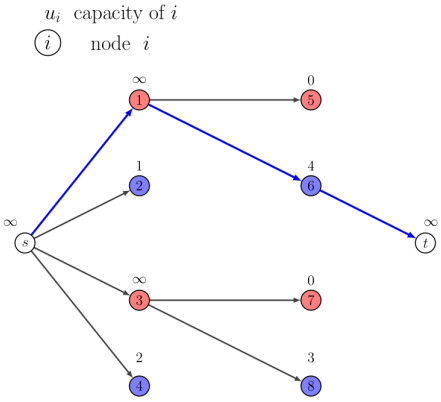
For every value of l_1, the algorithm associates to each node $i \in \mathcal{N}$ a capacity u_i according to Equation (14); for $l_1 = 2$, the algorithm obtains the network in Figure 1(b). Then, the algorithm solves the maximum capacity path problem. At termination, Algorithm 3 provides a tree of maximum capacity paths rooted in s, and $v^* = 4$ is the capacity of the s-t path



(a) Heterogeneous s-t network.



(b) Network for $l_1 = 2$.



(c) Maximum capacity tree

Figure 1: Computation of row of survival signature corresponding to $l_1 = 2$ using the SO method.

in this tree. Therefore, for any $l_2 > l_2^*$ $(= n_2 - v^* = 4 - 4 = 0)$, the network is up. The structure function evaluation resulting from P_1 and P_2 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 analyze the complexity of the SO approach. Using a binary heap, operations insert, increase-key, and delete-max require $O(\log n)$ time, while the other operations require $O(1)$ time [59]. The initialization portion of SO-MaxCapPath algorithm requires $O(n)$ time; operations find-max and delete-max may be performed n times for a total time complexity of $O(n \log n)$; the operations in lines 10-11 and 14-15 may be performed, in the worst case, for every arc in the network, where the assignment operations (lines 10 and 14) have $O(1)$

Table 1: Structure function for permutations $P_1 = \{5, 7, 3, 1\}$ and $P_2 = \{2, 4, 8, 6\}$ computed with the SO method.

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

complexity and the heap operations have $O(\log n)$ complexity, and hence the overall complexity of the while loop is $O(m \log n)$. Therefore, the overall complexity of Algorithm 3 is $O(m \log n)$. In Algorithm 4, the dominating factor is the For loop over n_1, and for every iteration of the loop, the algorithm solves a maximum capacity path problem using Algorithm 3; the algorithm repeats this process M times. Therefore, the overall time complexity of the single-objective optimization approach is $O(n_1 m \log(n) M)$.

3.3 Multi-Objective Optimization Approach

In comparing the two previous approaches, although the SO method has the same worst-case complexity as the Naive approach with bisection search, one can conjecture that the SO approach tends to be faster because it avoid some of the unnecessary work performed by the Naive approach. For instance, the Naive algorithm runs a new BFS for every combination of n_1 and n_2, completely disregarding the information obtained in the previous BFS run. By contrast, the SO algorithm is able to reduce much of this effort by solving an optimization problem with a slightly worse complexity than the one required by a BFS, but for every row of the survival signature instead of for every entry of the matrix. The multi-objective optimization approach (MO) extends the idea of the SO approach by solving a single multi-objective optimization problem (instead of $n_1 + 1$ single-objective optimization problems) to evaluate the structure function matrix (e.g., in Table 1) in each replication.

As with the Naive and SO 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 (10). To every node $i \in \mathcal{N}$, associate two weights according to

$$u_i^e = \begin{cases} q_i^e, & i \in \mathcal{N}_e, \\ \infty, & i \in \mathcal{N}/\mathcal{N}_e, \end{cases} \qquad e = 1, 2. \tag{17}$$

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

In order to compute the survival signature for the jth replication, we must determine all combinations of l_1 and l_2 for which the network is down. In the MO approach, we determine these combinations by solving the bi-objective maximum capacity path problem, 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

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

Similarly to the SO approach, these capacities represent, respectively, the number of node failures in $\mathcal{N}_1$ and in $\mathcal{N}_2$ that would destruct the path p, that is, it would be necessary $c^1(p)$ failures in $\mathcal{N}_1$ or $c^2(p)$ failures in $\mathcal{N}_2$ to destruct the s-t path p.

Define a path p from s to i as a non-dominated 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 point as the image of a non-dominated path in the objective space. Then, the bi-objective maximum capacity path problem can be defined as

$$\max_{p \in \mathcal{P}} \{c^1(p), c^2(p)\}, \tag{18}$$

and a solution to this problem provides a set Ω of non-dominated points (v_1^*, v_2^*), which can be used to represent 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 the $\Psi(\mathbf{x}(l_1, l_2)) = 1$. 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, $\Psi(\mathbf{x}(l_1, l_2)) = 1$. □

Bi-objective and multi-objective shortest path problems have been studied since the 1980s with [60] and [61], and some of the more recent works in this area includes [14, 62, 63].

We build our MO-MaxCapPath algorithm (given in Algorithm 5) by modifying the BDijkstra algorithm essentially in the same way we modified the Dijkstra's algorithm for the shortest path problem to solve the single-objective maximum capacity path problem in the SO approach. In the MO approach, 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 potencial non-dominated path from node s to node i. 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 3. After it is possible to conclude that a candidate s-i path

p is non-dominated, the values $(\mathsf{d}_i^1, \mathsf{d}_i^2)$ are made permanent, 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 (16).

The correctness of the BDijkstra algorithm has been proved for the bi-objective shortest path problem in [14]. Because our extension of this algorithm is analogous to the extension of Dijkstra's algorithm from (single-objective) shortest path to maximum capacity path, we have chosen not to prove it again here. Instead, we explain the steps of BDijkstra in the context of the bi-objective maximum capacity path problem and provide an empirical validation of the algorithm in the next section.

Algorithm 5: MO-MaxCapPath

```
▷ Set N_i = 0, d_i^1 = 0, d_i^2 = 0, InH[i] = False, i ∈ 𝒩 \ {s}
▷ Set N_s = 0, d_s^1 = ∞, d_s^2 = ∞, l = (s, ∞, ∞, −, −)
insert(l, H); InH[s] = True
while H ≠ ∅ do
    l* = find-max(H), delete-max(H)
    d_{i*}^1 = 0, d_{i*}^2 = 0                          // i* is the node with label l*
    N_{i*} = N_{i*} + 1, L[i*][N_{i*}] = l*, InH[i*] = False
    l^new = NewCandidateLabel(i*, l*)
    if l^new ≠ Null then
        insert(l^new, H), InH[i*] = True
        d_{i*}^1 = l^new.d^1, d_{i*}^2 = l^new.d^2
    end
    RelaxationProcess(i*, H, l*)
end
return L[t]
```

Following the notation of [14], the data structure used in Algorithm 5 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 $\mathsf{L}[i][1]$, $\mathsf{L}[i][2], \ldots, \mathsf{L}[i][\mathsf{N}_i]$, which were stored in lexicographically decreasing order. Non-dominated points are represented by labels of the form

$$(i, \mathsf{d}^1, \mathsf{d}^2, j, r), \tag{19}$$

where i denotes the 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 [14], Algorithm 5 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. The candidate label of node i is not in $\mathsf{L}[i]$ since a label is stored in L only when the label becomes permanent. This is an important invariant in the method of [14], i.e., the label l in H associated with node i is not dominated by any label in $\mathsf{L}[i]$, for all $i \in \mathcal{N}$. Additionally, 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 same basic operations previously discussed, but now 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, 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 $l^*.\mathsf{d}^2 > l.\mathsf{d}^2$.

The label l^* extracted from the heap in an iteration becomes permanent and is added to $\mathsf{L}[i]$ since (1) l^* is not dominated by any label in $\mathsf{L}[i]$ according to the mentioned invariant, and (2) there is no non-explored path from s to i whose pair of capacities dominates $(\mathsf{d}^1, \mathsf{d}^2)$. Item (2) is proved in [14]. Then, the non-dominated (permanent) labels of any node are determined in lexicographic decreasing order, and once l^* is extracted from the heap, it is added to the end of $\mathsf{L}[i]$. With this, in the dominance test it is only necessary to check the if last label in

$\mathsf{L}[i]$ dominates the l^*. Once a label l^* associated with a node i is extracted from the heap, the algorithm must check if other labels associated with successors of i can be improved. This is accomplished with the relaxation process (see procedure RelaxationProcess). Observe that in the dominance test, the relaxation process only checks if a candidate label associated with node j is not dominated by the last label inserted into $\mathsf{L}[j]$.

Solving the bi-objective maximum capacity path problem provides us with the set of non-dominated points Ω. Then, we can populate Φ_j 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 6 states the MO approach.

We now establish the complexity of Algorithm 5. 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 delete-max operation in $O(\log n)$, a series of assignment operations each having complexity $O(1)$, and, when the label returned by 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 NewCandidateLabel and 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 $O(nn_1 \log n)$ plus the complexity of NewCandidateLabel and RelaxationProcess. In every iteration, function 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 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 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 $\sum_{i \in \mathcal{N}} |\Gamma_i^-| = m$.

```
1  Function NewCandidateLabel(i*, l*):
2      d^1 = 0, d^2 = 0; l^new = Null          // Γ_i^- is the set of predecessors of node i
3      for j ∈ Γ_i^- do
4          for l ∈ L[j] do
5              f^1 = min{l.d^1, u^1_{i*}, u^1_j}
6              f^2 = min{l.d^2, u^2_{i*}, u^2_j}
7              if f^1 > d^1 or f^1 == d^1 and f^2 > d^2          // lexmax cand label
8               then
9                  if f^1 < l*.d^1 and f^2 > l*.d^2          // non-dom cand label
10                  then
11                     d^1 = f^1;  d^2 = f^2
12                     l^new = (i*, d^1, d^2, j, r)          // r is the position of l in L[j]
13                 end
14             end
15         end
16     end
17     return l^new
```

```
1  Procedure RelaxationProcess(i*, H, l*):
2      for j ∈ Γ_i^+ do
3          f^1 = min{l*.d^1, u^1_{i*}, u^1_j}
4          f^2 = min{l*.d^2, u^2_{i*}, u^2_j}
5          if f^1 > d^1_j or f^1 == d^1_j and f^2 > d^2_j          // Relaxation (i*, j)
6           then
7              if N_j == 0 or f^1 < L[j][N_j].d^1 and f^2 > L[j][N_j].d^2      // non-dom. label
8               then
9                  d^1_j = f^1; d^2_j = f^2
10                 l = (j, d^1_j, d^2_j, i*, N_{i*})          // N_{i*} is the position of l* in L[i*]
11                 if InH[j] == False then
12                     insert(l, H)
13                     InH[j] = True
14                 end
15                 else
16                     increase-key(l, H)
17                 end
18             end
19         end
20     end
```

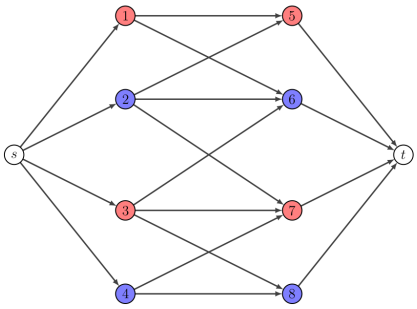
Algorithm 6: MO-SurvSig

$\Phi \leftarrow \mathbf{0}$
for $j = 1$ **to** M **do**
  ▷ Simulate a permutation of $\mathcal{N}_1$ and a permutation of $\mathcal{N}_2$
  ▷ Update $u_i,\ i \in \mathcal{N}$, according to Equation (17)
  $\Omega =$ MO-MaxCapPath($\mathcal{G}$) // Ω stores all (v_1^*, v_2^*) points
  for $(v_1^*, v_2^*) \in \Omega$ **do**
    for $l_1 = 0$ **to** n_1 **do**
      for $l_2 = 0$ **to** n_2 **do**
        if $v_1^* > n_1 - l_1$ **and** $v_2^* > n_2 - l_2$ **then**
          $\phi(l_1, l_2) \leftarrow \phi(l_1, l_2) + 1$
        end
      end
    end
  end
end
$\Phi = \Phi/\mathsf{M}$.
return Φ

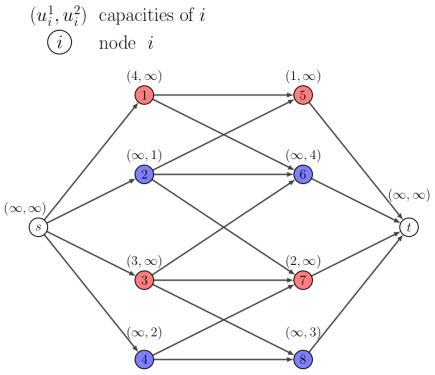
Each iteration, for every $j \in \Gamma_i^+$, RelaxationProcess performs a sequence of constant time operations and either an insert or increase-key operation in the worst case with complexity $O(\log n)$. Like the NewCandidateLabel, RelaxationProcess is called N_i times for each $i \in \mathcal{N}$, which amounts to a total complexity of $O(\sum_{i\in\mathcal{N}} \left(\mathrm{N}_i \log n|\Gamma_i^+|\right))$. 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 5 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 the MO approach is therefore $O\left((n_1^2 m + n_1 m \log n)\mathsf{M}\right)$.

We now illustrate the MO approach with the following example. 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 (17) (see Figure 2(b)), and solves the corresponding bi-

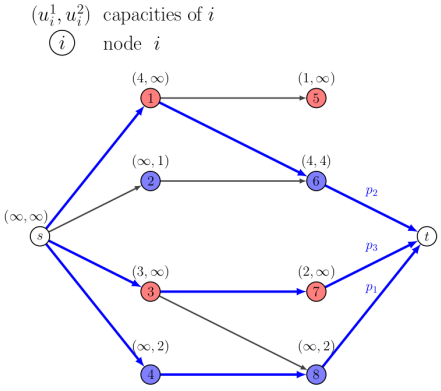
objective maximum capacity path problem (see Figure 2(c)). The solution of the bi-objective maximum capacity path problem shows that there are three non-dominated paths from s to t: $p_1 = s$ - 4 - 8 - t, $p_2 = s$ - 1 - 6 - t, and $p_3 = s$ - 3 - 7 - t, with respective non-dominated points $(\infty, 2)$, $(4, 4)$, and $(2, \infty)$; these are the points stored in Ω. The algorithm then loops over the these points populate 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 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.



(a) Heterogeneous s-t network.



(b) Network after the updating of u_i.



(c) Maximum capacity path graph

Figure 2: Computation of survival signature using MO approach.

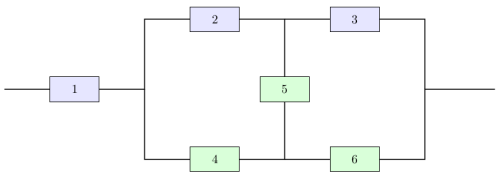
Table 2: Structure function for permutations $P_1 = \{5, 7, 3, 1\}$ and $P_2 = \{2, 4, 8, 6\}$ computed with the MO method.

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

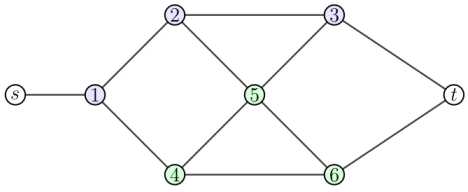
4 Computational Experiments

4.1 Validation of Approaches

First, we validate the Naive approach by verifying that the survival signature obtained by it when all permutations of $\mathcal{N}_1$ and $\mathcal{N}_2$ are considered is the exact survival signature. For this, we compare the survival signature obtained through the Naive approach with the exact survival signature for a small bridge system obtained from [58]. The reliability block diagram (RBD) of this system is depicted in the Figure 3(a). This system is composed of six elements subject to failure, numbered 1 through 6. For this system, $\mathcal{N}_1 = \{1, 2, 3\}$, and $\mathcal{N}_2 = \{4, 5, 6\}$. We represent this system as a two-terminal network in Figure 3(b), where we appended artificial nodes s and t. The analytical value of the survival signature is also provided in [58] and can easily be verified.



(a) Reliability block diagram of the first system used for validation. The network was obtained from the paper by [58].



(b) Network representation of validation system with nodes s and t appended.

Figure 3: Bridge system and corresponding s-t network

Table 3 presents the analytical survival signature versus the survival signature obtained through the Naive approach when we set Algorithm 2 to run all 36 possible combinations of

$\mathcal{N}_1$ and $\mathcal{N}_2$. From Table 3 it is clear that the survival signature obtained through the Naive approach is indeed the exact survival signature of the system.

Table 3: Analytical survival signature versus survival signature obtained through Naive approach for the bridge system of Figure (3).

(a) Analytical survival signature.

$l_1 \backslash l_2$	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

(b) Exact survival signature with Naive approach.

$l_1 \backslash l_2$	0	1	2	3
0	0	0	0	0
1	0	0	0.111111	0.333333
2	0	0	0.222222	0.666666
3	1	1	1	1

The next step in our validation process is to verify the consistency of the survival signature obtained by the three methods; for this, consider the network of Figure (4). This network is composed by $n = 20$ nodes and $m = 30$ arcs. Nodes s, 6, 7, 10, 12, 13, 17, and t are completely reliable, while the remaining nodes are subject to failure and can be grouped into two groups according to their failure probability distribution. Nodes 1, 3, 5, 8, 14, and 16 have failure probability distribution $F_1(t)$ and nodes 2, 4, 9, 11, 15, and 18 fail according to $F_2(t)$.

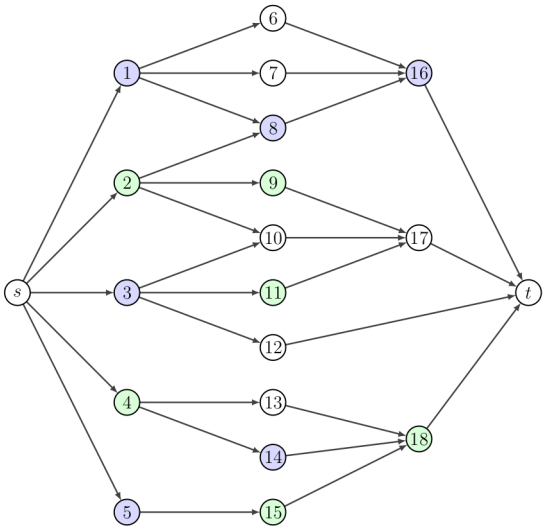


Figure 4: Network used to compare the results from the three methods.

Observe that since we have six elements in each group, it would be necessary to generate

$6! \times 6! = 518400$ permutations in order to compute the exact survival signature, which would already require a considerable computational effort for such a small example. We estimated the survival signature using the three approaches with $\mathsf{M} = 50000$ replications. We set the same seed to the random number generator used by all the algorithms so that the three algorithms generate the same permutations in the same order. The results from the algorithms are equal and the estimated survival signature for this system is shown in Table 4.

Table 4: Survival signature estimate obtained by the three methods using the same random sequences

$\hat{\Phi}$

$l_1 \backslash l_2$	0	1	2	3	4	5	6
0	0	0	0.0665	0.20016	0.39894	0.49952	1
1	0.16728	0.16728	0.20056	0.31688	0.49886	0.63234	1
2	0.40106	0.40106	0.40106	0.50076	0.65582	0.79968	1
3	0.64976	0.64976	0.6662	0.69962	0.8168	0.93396	1
4	0.86712	0.86712	0.87852	0.90054	0.9339	1	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1

4.2 Comparison of Approaches

We now proceed to discuss the results of our computational experiments. All experiments were accomplished with C++ on an Intel® Core i7-4500U CPU with 4×1.8 GHz processor and 8 GB RAM running on an Ubuntu 20.04 OS. In these experiments we considered the random geometric graph (RGG) shown in Figure (5). This network contains 350 nodes, including 149 in $\mathcal{N}_1$ (marked in blue), 199 in $\mathcal{N}_2$ (marked in red), and nodes s and t. For both classes of nodes, the failure distribution is a Weibull with scale parameter $\lambda = 100$. The failure distribution $F_1(t)$ has a shape parameter $\beta_1 = 2.5$ while the failure distribution $F_2(t)$ has a shape parameter $\beta_2 = 3.0$. This network was created according to the following procedure: 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 $i \in \mathcal{N}$, randomly generate coordinates x_i and y_i

between 0 and 10 according to a uniform distribution, and create and arc from node i to node $j \in \mathcal{N}$ if and only if $x_i \leq x_j$ and the Euclidean distance between i and j is smaller than or equal to 2. For this example, the procedure generated $m = 3231$ arcs.

We estimated the survival signature and the reliability (see Figure (6) for $\mathsf{M} = 1000$) of this network using the three proposed approaches with the same seed for the random number generator and for different numbers of replications: $\mathsf{M} = 1000$, 10000, and 50000, but with a processing time limit of six hours, after which we censored the experiment. Tables 5–7 show the results. In each table, the first column indicates the number of replications of the experiment, the second column indicates the total processing time (which includes initialization and network generation, survival signature estimation, and reliability estimation), the third column indicates the survival signature estimation time and the corresponding percentage with respect to the total time, and the last column indicates the average time per replication.

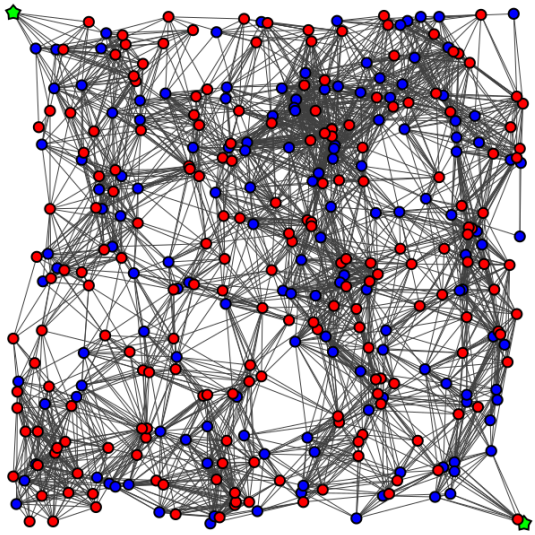


Figure 5: Network with $n = 350$, where $n_1 = 149$, $n_2 = 199$, and $m = 3231$ arcs.

Table 5: Performance analysis of the Naive approach for 1000, 10000, and 50000 replications.

M	Total Time (sec)	Surv. sig. estim. (sec)/(%)	Time per replic. (sec)
1000	3537.398	3533.8 (99.9%)	3.53
10000	n/a	n/a	n/a
50000	n/a	n/a	n/a

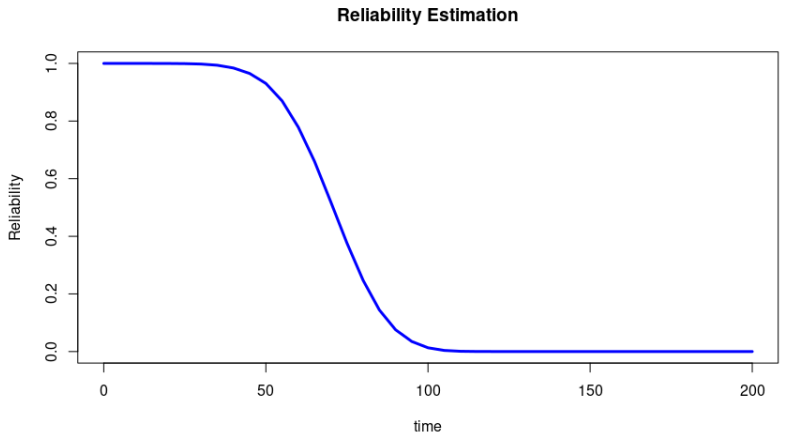


Figure 6: Reliability estimated with the three methods for MO = 1000 replications (blue), and reliability curves (red) calculated with the 95% upper and lower bound on the survival signature.

Table 6: Performance analysis of the SO approach for 1000, 10000, and 50000 replications.

M	Total Time (sec)	Surv. sig. estim. (sec)/(%)	Time per replic. (sec)
1000	1043.65	1041 (99.7%)	1.04
10000	10716.031	10683 (99.69%)	1.07
50000	n/a	n/a	n/a

Table 7: Performance analysis of the MO approach for 1000, 10000, and 50000 replications.

M	Total Time (sec)	Surv. sig. estim. (sec)/(%)	Time per replic. (sec)
1000	5.31	3.228 (60.79%)	3.228×10^{-3}
10000	30.829	28.725 (93.17%)	2.872×10^{-3}
50000	145.995	143.82 (98.51%)	2.876×10^{-3}

From Table 5, it follows that the survival signature estimation is indeed the bottleneck operation in estimating the two-terminal reliability through the Naive approach, corresponding to more than 99% of the processing time for only one thousand replications. The experiments with ten thousand and fifty thousand replications were censored after six hours, but since the algorithm processing time increases approximately linearly with the number of replications, one can estimate the time needed for the algorithm to complete the censored experiments. The Naive-SurvSig would take approximately ten hours to complete the 10000 replication experiment, and fifty hours to complete the 50000 replication experiment. Tables 6 and 7 show that the SO approach is indeed faster than the Naive approach (approximately 3

times faster), and the MO approach is the fastest approach, more 1000 times faster than the Naive and more than 300 times faster than the SO approach. By the complexity analysis in Section 3.1, we anticipate that using Naive-bisection would be faster than Naive, but not by more than a factor of n_2. Thus, we would expect the time per iteration for Naive-bisection to be no better than $3.53/199 \approx 1.77 \times 10^{-2}$ seconds per replication, which is still substantially slower than the MO approach.

We have performed more experiments with the MO algorithm using the same network but substantially increasing the number of replications. We have solved this problem for 100000, 500000, and 1000000 replications. The results show that the algorithm scales very well and maintains an approximately constant time per iteration of 2.8×10^{-3} seconds. This allowed us to run the 1000000 replications in approximately 50 minutes. The MO approach can therefore be used to solve two-terminal reliability problems in large, heterogeneous networks with a large number of replications to improve accuracy, which would not be possible with the other two approaches.

5 Considerations and Future Research

In this paper, we proposed a MC-signature method (which we refer to as the MO approach) to estimate the survival signature and the two-terminal reliability of networks with heterogeneous components based on solving a multi-objective optimization problem. To the best of our knowledge, this is the first work to point out the relationship between the multi-dimensional survival signature computation and a multi-objective optimization problem. We discussed two other approaches to estimate the two-terminal reliability: the Naive approach based on performing breadth-first search, and the SO approach based on solving a single-objective optimization problem. These three approaches are equivalent in the sense that they generate the same output but require different amounts of time. The computational complexity of the three approaches are shown in Table 8.

We validated the three approaches for consistency by solving the same problem with the

Table 8: Complexity of Naive, SO, and MO approaches.

Approach	Complexity
Naive	$O(n_1 n_2 m\mathsf{M})$
Naive-bisection	$O(n_1 m \log(n)\mathsf{M})$
SO	$O(n_1 m \log(n)\mathsf{M})$
MO	$O\left((n_1^2 m + n_1 m \log n)\mathsf{M}\right)$

three methods generating the same random permutations, which produced the same result for the three approaches. Then we performed computational experiments to compare the performance of the methods. The MO method proved to be the fastest and to scale well as we increase the number of replications. The experiments demonstrated that the MO can offer an efficient way of estimating the two-terminal reliability for large, complex, heterogeneous networks. As immediate future research directions, we consider the following topics:

1. The literature provides opportunities for enhancements in terms of worst case complexity for the MO approach; for instance, Sedeno-noda and Colebrook [14] presented an improvement to the NewCandidateLabel that would reduce its complexity to $O(n_1 m)$, presumably reducing the overall complexity of our MO approach to

$$O\left((n_1 m + n_1 m \log n)\mathsf{M}\right) = O(n_1 m \log(n)\mathsf{M}),$$

   which would equal the complexity of the SO and Naive-bisection approaches.

2. Other implementations of Dijkstra's algorithm, such as Dial's implementation, may be able to yield better empirical results for the SO problem. A Dial's implementation of the MO problem is also possible, but the number of buckets (and thus the memory and time complexity) appears to increase substantially in this case.

3. It may be possible to improve the Naive or Naive-bisection based on realizing that l_2^* (i.e., the maximum value of l_2 for which $\Psi(\mathbf{x}(l_1, l_2)) = 0$ for fixed l_1) must be non-increasing as l_1 increases; thus, if l_2^* is obtained according to Equation (13) for row l_1, the search in row $l_1 + 1$ need only consider $l_2 = 0, 1, \ldots, l_2^*$. Also, it may be possible to

improve SO approach by using one iteration's permutation labels to initialize the next iteration.

4. More computational experiments are needed to investigate the algorithm's efficacy on larger networks, special types of networks, and different ratios of n_1/n_2.

Finally, we are also contemplating the possibility of generalizing this work to more than two classes of nodes, as well as to other reliability problems, such as K-terminal or coverage reliability.

References

[1] 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.

[2] 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.

[3] 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.

[4] W.-C. Yeh, Y.-C. Lin, and Y. Y. Chung. Performance analysis of cellular automata Monte Carlo simulation for estimating network reliabilty. *Expert Systems with Applications*, 37:3537–3544, 2010.

[5] F.J. Samaniego. On closure of the IFR class under formation of coherent systems. *IEEE Transactions on Reliability*, R-34:69 – 72, 05 1985.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] A. P. Punnen. A linear time algorithm for the maximum capacity path problem. *European Jornal of Operational Research*, 53:402–404, 1991.

[14] A. Sedeño-noda and M. Colebrook. A biobjective Dijkastra algorithm. *European Journal of Operational Research*, 276, 2019.

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

[16] 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.

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

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

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

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

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] J. E. Ramirez-Marquez and C. M. Rocco. Evolutionary optimization technique for multi-state two-terminal reliability allocation in multi-objective problems. *IIE Transactions*, 42:539–552, 2010.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

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

[35] 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.

[36] 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.

[37] I. Wegener. BDDs - design, analysis, complexity, and applications. *Discrete Applied Mathematics*, 138:229–251, 2004.

[38] A. Ferrara, P Liberatore, and M. Schaerf. The size of BDDs and other data structures in tempora logics model checking. *IEEE Transactions on Computers*, 65(10):3148–3156, 2016.

[39] 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.

[40] L. Gertsbakh and Y. Shpungin. *Network Reliability and Resilience*. Springer Briefs in Electrical and Computer Engineering. Springer, 2011.

[41] 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.

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

[43] 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.

[44] Y. Shpungin. Networks with unreliable nodes and edges: Monte Carlo lifetime estimation. *World Academy of Science, Engineering and Technology*, 3, 2007.

[45] 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.

[46] J. Navarro and T. Rychlik. Reliability and expectation bounds for coherent systems with exchangeable componenets. *Journal of Multivariate Analysis*, 98:102–113, 2007.

[47] 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, 2007.

[48] 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.

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

[50] 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.

[51] I. Gertsbakh, Y. Shpungin, and F. Spizzichino. Two-dimensional signatures. *Journal of Applied Probability*, 49:416–429, 2012.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] S. Eryilmaz and A. Tuncel. Generalizing the survival signature to unreparable homogeneous multi-state systems. *Naval Research Logistics*, 63, 2016.

[57] 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.

[58] 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.

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

[60] P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making Theory and Application*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*. Springer, 1980.

[61] E. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16:236–245, 1984.

[62] S. Demeyer, J. Goedgebeur, P. Audenaert, M. Pickavet, and P. Demeester. Speeding up Martin's algorithm for multiple objective shortest path problems. *4OR - A Quarterly Journal of Operations Research*, 11:323–348, 2013.

[63] D. Duque, L. Lozano, and A. L. Medaglia. An exact method for biobjective shortest path problem for large-scale road networks. *European Journal of Operational Research*, 242:788–797, 2015.