1-2020

Scalable, adaptable and fast estimation of transient downtime in virtual infrastructures using convex decomposition and sample path randomization

Zhiling GUO
Singapore Management University, ZHILINGGUO@smu.edu.sg

Jin LI

Ram RAMESH

Published in INFORMS Journal on Computing, 2099, Advance online.
https://doi.org/10.1287/ijoc.2019.0888

Accepted version

Scalable, Adaptable and Fast Estimation of Transient Downtime in Virtual Infrastructures using Convex Decomposition and Sample Path Randomization

Zhiling Guo
School of Information Systems, Singapore Management University, Singapore 178902,
zhilingguo@smu.edu.sg

Jin Li
School of Economics and Management, Xidian University, Xi'an, China 710071,
jinli@xidian.edu.cn

Ram Ramesh
Department of Management Science and Systems, State University of New York, Buffalo, NY 14260,
rramesh@buffalo.edu

Network function virtualization (NFV) enables efficient cloud resource planning by virtualizing network services and applications into software running on commodity servers. A cloud service provider needs to manage and ensure service availability of a network of concurrent virtualized network functions (VNFs). The downtime distribution of a network of VNFs can be estimated using sample path randomization on the underlying birth-death process. An integrated modeling approach for this purpose is limited by its scalability and computational load due to the high dimensionality of the integrated birth-death process. We propose a generalized convex decomposition of the integrated birth-death process, which transforms the high-dimensional multi-VNF process into a series of interlinked low-dimensional single-VNF processes. We theoretically show the statistical equivalence between the transition probabilities of the integrated birth-death process and those resulting from interlinking the decomposed system of processes. We further develop a decomposition algorithm that yields scalable and fast estimation of the system downtime distribution. Our algorithmic framework can be easily adapted to any logical definition of overall system availability. It can also be easily extended to various realistic VNF network configurations and characteristics including heterogeneous VNF failure distributions, effects of both node and link failures on the overall system downtime of fully or partially connected networks, and resource sharing across multiple VNFs. Our extensive computational results demonstrate the computational efficiency of the proposed algorithms while ensuring statistical consistency with the integrated network model, and the superior performance of the decomposition strategy over the integrated modeling approach.

Key words: cloud computing; convex decomposition; Markov chains; network virtualization; sample path randomization

1. Introduction

Different from traditional networks that consist of a large variety of dedicated elements such as routers, firewalls and gateways, *virtual infrastructure* (VI) is a software-defined IT infrastructure running on top of a physical substrate to provide both computing and communication as a service. Within the framework of VI, *Network Function Virtualization* (NFV) is a fast emerging concept, where network functions traditionally carried out by proprietary and dedicated hardware are virtualized as software-defined functions hosted on virtual machines on standard server platforms. With the exponential growth in connected devices, NFV has gained broad industry traction. A recent Infonetics report forecasts the global NFV market to grow more than 5-fold and reach $11.6 billion by 2019 (IHS 2015). If successfully developed and implemented, NFV will completely revolutionize how networks are built, managed and used to create services.

1.1. NFV Architecture and Deployment

NFV aims at decoupling network functions from specialized hardware through virtualization technology. The virtualized software implementation of a network function is called *virtualized network function* (VNF). When a router is virtualized, it is called router VNF. When a base station is virtualized, it is called base station VNF. Other examples of VNFs include session border controllers, load balancers, firewalls, etc (ETSI 2014). Figure 1 illustrates the NFV architecture comprising of NFV infrastructure (NFVI) and VNF deployment (ETSI 2013a, Martini and Paganelli 2016, Mijumbi et al. 2016).

The NFVI includes both physical resources and virtual resources. The physical resources include computing hardware, storage and network that provide processing, storage and connectivity to VNFs. The virtual resources are abstractions of the computing, storage and network resources, which are managed using a virtualization layer based on a hypervisor. NFVI provides the hardware and software environment in which VNFs are deployed.

The VNF is the basic building block in the NFV architecture. The abstraction of network functions away from dedicated hardware allows VNFs to be hosted on standard server platforms in the cloud data center. In the data center environment, the virtual computing and storage resources may be represented in terms of one or more virtual machines (VMs), while virtual networks are made up of virtual links and nodes. A VNF can be deployed over multiple VMs. A VNF link can be used to connect between two or more VNFs.

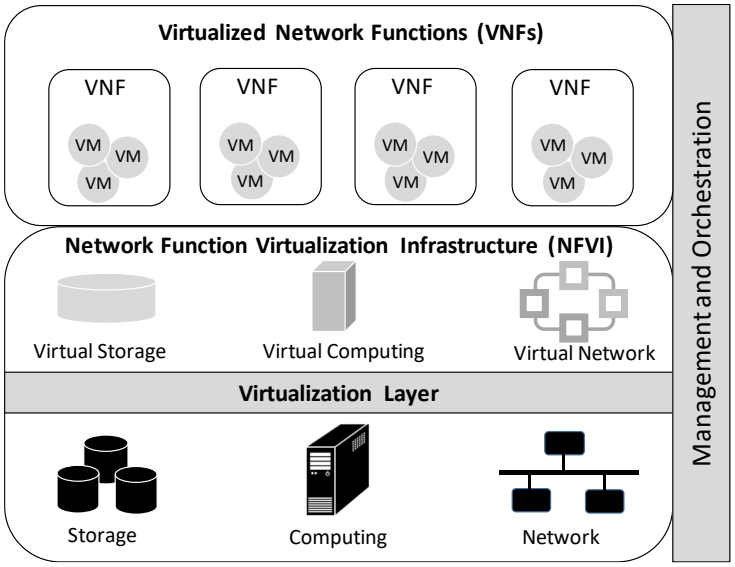

Figure 1 Network Function Virtualization Architecture

Similar to Software as a Service (SaaS) model in cloud computing, a network service can be provided by VNFs as an end-to-end service chain through virtualized infrastructure management and NFV orchestration. For example, multiple VNFs such as firewalls, load balancers and session border controllers need to be connected using virtual links to create an end-to-end service. The session border controller can be further split into different VNFs that are operated at different network locations: session terminations executed at the edge of the network, admission control executed in the core network, and statistics and billing data collection executed at a data center. A stream of network packets flow through the VNFs in sequence to complete an end-to-end service (Martini and Paganelli 2016). The ESTI Industry Specification Group for NFV provide a few other use cases (ETSI 2013b).

NFV has the promise to bring flexibility and significant cost savings to networking, which is essential to build future dynamic and service-aware networks (Sun et al. 2016). However, in the cloud environment, servers experience random and transient failures such as disk failures or software faults (Siewiorek and Swarz 2017). Such failures normally occur for short periods of time and the failures can be quickly recovered. Deployment of NFV in large-scale networks require the service providers to continuously monitor the VM failures in order to fulfill the uptime guarantee specified in the *service level agreement* (SLA) (Buyya et al. 2011). Transient analysis of the system downtime of a network of VNFs is thus critical in SLA-based resource provisioning for cloud computing. We are among the

first to propose efficient, flexible, scalable and extensible estimation algorithms to evaluate the availability and performance of VNFs within the NFV environment.

1.2. Research Scope

In this study, we focus on scalable, adaptable and fast estimation of the transient downtime distribution for a set of interconnected VNFs in a data center or a network of data centers. We assume the network consists of M VNF nodes and L communication links that constitute an end-to-end service chain. We are interested in estimating the overall availability of the service chain under transient conditions. Service availability is specified as an uptime guarantee in an SLA between a cloud service provider and its client as follows: In the basic form, the client requires n_m VMs for each VNF, $m = 1, \ldots, M$, during a service window of length T to be concurrently available at a guaranteed percentage of uptime. In the general form, this requirement could be specified over any logical combination of the M VNFs to be available at the guaranteed level. To ensure VNF availability, the service provider usually assigns k_m, $m = 1, \ldots, M$, additional VMs as backup. We refer to the n_m VMs as *primary* and the k_m VMs as *backup* VMs. To ensure availability of the network connectivity required by the client, the service provider can also provide backups for communication links. The need for the backups arises due to the random failure and repair processes experienced in the cloud environment. In this research, we focus on small-scale, transient failures such as isolated server/switch failures rather than large-scale failures such as rack/pod or even an entire data center outage (Gill et al. 2011). The backup VMs enable checkpointing by periodic capture of primary VM images to maintain continuity of the client's business when primaries fail and restarting the failed primaries when they return to service. Du et al. (2015) define and analyze three checkpointing strategies for this purpose: *Powered-on without delay*, *Powered-on with delay* and *Powered-off*. The proposed downtime estimation methodology can be used in any checkpointing strategy. Without loss of generality, we assume the most commonly used powered-on without delay checkpointing strategy in this research.

An SLA typically entails a penalty to the service provider if the guaranteed uptime is not met in the service period. The provisioning of the backups reduces the likelihood of downtime occurrence but entails extra resource costs. Thus, knowledge of the downtime probability distribution for any given network configuration is essential in the service provider's resource allocation. The service downtime distribution depends on (i) the notion

of VNF availability specified in the SLA, (ii) the underlying random failure and repair processes involved, and (iii) the checkpointing strategies used in resource management. We model the server failure and repair processes in any configuration of the overall system as a birth-death Markov process. The steady-state availability of the VNFs under any notion of availability can easily be obtained from such a process characterization. However, as Du et al. (2015) point out, the attainment of steady-state can be a strong assumption under realistic service windows commonly seen in practice. Further, the demonstration of a configuration reaching steady-state and remaining there throughout a contract period in practice could be very difficult. Hence, we assume transient conditions in deriving the downtime distribution of a VNF configuration.

We consider three types of VM failures that are commonly encountered in practice: *Exponential*, *Weibull* and *Erlang* failure distributions. The Exponential failure processes typically represent normal data center operations in commercial practice. The Weibull distribution captures aging and infant mortality systems, and the Erlang captures the partially failing systems. Compared to failure, the repair processes are more controllable and hence, were modeled using the Exponential distribution. Using server log data from the Center for Computational Research (CCR) at SUNY at Buffalo, we fitted the three failure distributions as well as the Exponential distribution for repairs to estimate their parameters after validating their distributional assumptions (see Du et al. 2015). The computational results presented in this paper are based on these parametric estimates.

Although each VM failure event is independent, the VNF failures may be correlated because some VM nodes can be shared across multiple VNFs. In addition to VNF failures, network availability is also affected by link failures. The service provider may provide both *link-level* and *path-level* backups to guarantee the VNF network connectivity. The link-level backup refers to redundant links between any pair of VNFs that require connectivity. The path-level backup is achieved by alternative network paths that indirectly connect a pair of VNFs when their existing direct connectivity fails. In summary, we consider in this study the following realistic system configurations and characteristics: heterogeneous VM failure distributions, VM sharing across multiple VNFs, both VNF node and link failures that affect network availability, and different node and link backup strategies for fully-connected as well as partially-connected VNF networks.

1.3. Research Approach

The proposed estimation methodology is based on our earlier work on the *sample path randomization* (SPR) approach to estimating transient downtime in a single VNF configuration presented in Du et al. (2015). While a direct extension of this methodology to the multiple VNFs case is theoretically possible, such an approach suffers from the following major drawbacks. First, the sample path approach requires a full definition of the underlying birth-death process; as M and (n_m, k_m), $m = 1, \ldots, M$, vary, the structure of the underlying Markov process that collectively integrates the birth-death events in the full VNF configuration would also vary. Second, as the notion of availability specified in an SLA varies, the distinction between the down and up states of a configuration would also vary. Third, as the checkpointing strategies vary, the direct extension would require a complete redefinition of the underlying birth-death process states and transitions. All of these together lead to major problems of scalability and adaptability of a direct algorithm when extending single VNF model to any general VNF configuration by this integrated approach.

Motivated by these drawbacks, we develop an efficient strategy that yields fast, modular, reusable, scalable and adaptable algorithms to estimate the downtime distribution of multiple VNFs with statistical consistency. The proposed algorithmic strategy is summarized as follows. First, we decompose the integrated birth-death process into M independent birth-death processes, each pertaining to its corresponding VNF. We develop the concept of *convex decomposition* of birth-death processes for this purpose. Second, a randomized sample path for the integrated process configuration is obtained as an integration of independently generated sample paths that are randomized across the M processes. We establish the theoretical support for this strategy, and the statistical consistency between the downtime distributions obtained through the integrated model and the decomposed model, respectively. For the sake of simplicity of presentation and without loss of generality, we present the proposed algorithmic strategy using the basic form of availability where all VNFs are concurrently required to be available for overall system availability, and the powered-on without delay model of checkpointing. When these parameters change, the structure of the proposed algorithms will still remain the same.

1.4. Research Contributions

Our main contribution in this research is that we propose a generalized convex decomposition strategy that transforms the estimation of high-dimensional multi-VNF transient downtime distribution into the estimation of a series of interlinked low-dimensional single-VNF processes. We theoretically show the statistical equivalence between the transition probabilities of the integrated birth-death process and those resulting from the decomposed system. Our extensive computational results show that the proposed algorithms are computationally efficient while ensuring statistical consistency between the integrated and decomposed network model.

The convex decomposition approach yields fast and efficient estimation algorithms that are easily adapted to any logical definition of overall system availability. There are several other important advantages. First, due to the decomposable structure and low dimensionality, our algorithm is highly scalable to solve large network problems. Second, because each VNF independently generates its own sample path, the sample paths generated by the decomposed approach are reusable for other logical definitions of network availability. The modular structure enables the existing network model to be reused as building blocks for more complex services. Third, our algorithm can easily be extended to accommodate practical system requirements such as heterogeneous failure distributions, VM sharing across VNFs, and networks with various degrees of connectivity. In addition, the proposed decomposition framework is generalizable to the hierarchical network structure consisting of power nodes, switch nodes and virtual server nodes. Because the network dependence and independence can be modeled as AND/OR logical relationships, logical definition of network availability can be adapted based on different network topologies.

The rest of the paper is organized as follows. Section 2 reviews the relevant literature. Section 3 describes the sample path randomization strategy for the multiple VNF configurations. Section 4 develops the convex decomposition concept and its underlying theory. Section 5 presents the algorithmic implementation framework of the decomposition strategy. Section 6 summarizes our main experimental designs and computational analyses. Section 7 further discusses several algorithmic extensions and results. Finally, Section 8 presents the conclusions and directions for future research. A description of the notations is presented in the Appendix and the online supplement provides more detailed computational results of the analysis.

2. Related Literature

Virtual infrastructures comprising of high-speed optical networks spanning several distributed data centers have become predominant over the last several years (Mu et al. 2015, Chowdhury and Boutaba 2010). In particular, management of computing requests in such distributed environments has been the focus of several streams of research studies. The specific problems addressed in this domain include computing and communication resource allocation in the virtualized distributed environment (Sharkh et al. 2013), design of survivable VI components such as facility nodal servers and network links (Yu et al. 2012), mapping of VI requests to a substrate in a resource-efficient manner (Yu et al. 2010) and survivable mapping to ensure fault-tolerant service protection at the facility nodes and substrate nodes and links (Xiao et al. 2013, Yu et al. 2011).

In the context of VI, an infrastructure provider owns and maintains substrate networks, while a service provider assembles virtual networks from one or multiple infrastructure providers and provide end-to-end services to the users. Network function virtualization is a new emerging paradigm where network functions can be virtualized and chained together to provide the required functionality (Han et al. 2015). By decoupling network functions from the underlying proprietary hardware, the network functions can be executed on standard IT platforms like servers, switches and storage, and instantiated in various locations such as data centers, network nodes, and end-user premises. It opens the door for infrastructure sharing and automation on the cloud in a cost-effective way (Chowdhury and Boutaba 2010). When the VNFs are deployed on VI, managing the network infrastructure requires mapping of resources that guarantees VNFs are executed on the allocated VMs managed by their hypervisors.

In addition to resource mapping, infrastructure in the cloud environment is subject to a wide range of failures from small-scale, frequent failures to large-scale, less frequent failures. The types of failures can be hardware, software, human errors, hacks, and natural disasters (Gill et al. 2011). The failures result in service unavailability which affects the client applications running on the cloud infrastructure. In order to provide resilience to failure, service continuity and service assurance, the service provider typically adopts several checkpointing techniques to add fault tolerance into the computing systems (Lu et al. 2012). The checkpointing strategies ensure that the substrate can offer sufficient primary and backup resources to prevent VNF failures.

The literature on availability modeling in cloud data centers is still in its early stage of development. One of the earliest works that directly address the issue of quantifying availability in an Infrastructure-as-a-Service (IaaS) Cloud is Du et al. (2015). Their work emphasizes the efficient derivation of transient probabilities of service downtime. The advantage of transient probabilities lies in its applicability to finite and short-term service windows where steady-state is not guaranteed to be reached. Transient analysis of Markov chains is vital in availability modeling of service businesses, especially communication and queueing systems (Kaczynski et al. 2012).

While some studies use the terms reliability, survivability and availability interchangeably, there are significant distinctions between them. Reliability emphasizes exclusively on failures, as discussed in Fu (2010) where reliability is expressed as the time to failure and utilizing this metric, failure-aware node selection strategies are proposed. Studies on survivability, on the other hand, evaluate the ability of a VM to continue to provide services in an environment facing intrusion attacks and similar vulnerabilities, using methods such as continuous time Markov chains (Yang et al. 2013). In contrast, availability, as defined in Du et al. (2015) and as followed in this paper, incorporates both the failure and the repair processes and hence, the emphasis is on transient service downtime, which is a combined result of these two arrival processes over a period of time.

Existing research in service downtime focuses on defining various mechanisms to control this system metric. Some of these mechanisms study the VM migration process by changing the checkpointing protocols to ensure that downtime does not exceed a specified amount (Raad et al. 2014) and also by predicting failures caused by the checkpointing process itself (Lu et al. 2012). Configuring the minimum number of backup VMs is yet another tool to manage downtime, either by surviving a given number of host server failures (Machida et al. 2010) or by aiming to fulfill an SLA-specified availability guarantee with penalties for non-availability (Yuan et al. 2014). Our study derives an efficient algorithm to estimate the downtime distribution across multiple VNFs; this distribution may be used in many of the above-mentioned methods to manage service downtime in cloud data centers.

Sampling and simulation have been widely used to solve complex network problems. Parpas et al. (2015) employ importance sampling and Monte Carlo simulation to accurately estimate a high-dimensional function in solving multi-stage stochastic programming problems. This study builds upon the sample path randomization method proposed by

Silva and Gail (1986, 1989) and the successful application of the method in virtual server systems (Du et al. 2015). We propose a convex decomposition framework to efficiently estimate the transient downtime of a network of VNFs.

Our method has important implications in cloud computing, which aims to streamline the on-demand provisioning of resources as services through SLA management. Flexibility and scalability are the key design characteristics in an increasingly open cloud ecosystem. Passacantando et al. (2016) propose a scalable algorithm for multi-cloud resource provisioning in a distributed environment to ensure high quality of service. Research on the impact of VNF failures on the determination of SLA parameters such as uptime guarantee and price-penalty structures is still in its infancy. To the best of our knowledge, this research is the first approach to efficiently deriving the transient downtime distribution of a network of VNFs that would significantly impact the choice of these SLA parameters.

3. Sample Path Randomization

In the following discussion, we first summarize the sample path randomization strategy for the single VNF case developed in Du et al. (2015). Next, we present a direct extension of this approach to the multiple VNFs case. For notational simplicity, we denote these general configurations as M-VNF.

Consider a 1-VNF configuration comprising of n primary and k backup VM servers. Without loss of generality and for presentation simplicity, we assume one-to-one mapping between the virtual machines and physical machines as carried out in Du et al. (2015). We model the birth-death process for this configuration as follows: Each state of this process denotes the number of failed VM servers, and this ranges from 0 to $(n+k)$. Since the VNF is available if at least n VM nodes are concurrently available, the VNF is up in the states $0,\ldots,k$, and down in the states $(k+1),\ldots,(n+k)$. Next, we divide the service window of length T into $\lceil T/\Delta t\rceil$ intervals of length Δt. In a 1-VNF system, we denote the system state i_1^ρ as the number of failed servers at any given time interval indexed by $\rho = 1,2,\ldots,\rho_{max}$, where $\rho_{max} = \lceil T/\Delta t\rceil$. If Δt is sufficiently small, then within each time interval, either a server fails, a failed server is repaired, or the system remains in the same state. So the number of possible state transition within each time interval Δt is 3. It is important to note that the out-of-state transition probabilities are non-decreasing functions of Δt. The larger the time interval Δt, the more likely that we would observe a

server failure or repair event. A sample path of this discrete-time Markov chain comprises of a countably finite number of state transitions denoted as: $\{i_1^1, \ldots, i_1^{\rho_{max}}\}$. The transitions from each state of the sample path occur according to the transition probabilities in the next time interval Δt, and these probabilities are determined by the underlying VM failure and repair processes. Note that each sample path is a single realization of the evolution of the 1-VNF birth-death process during the service window T.

The total downtime, i.e. the total time the system experiences more than k VM failures during the service window T, is denoted by the random variable τ. The SPR algorithm in Du et al. (2015) derives the transient downtime distribution $v(\tau)$ using a sampling of sample paths approach and is shown in Algorithm 3.1. The strategy of this approach is summarized as follows. Starting from any initial state as the root, the set of all sample paths can be represented as a tree. The set of possible states to which a transition can occur from a given state are recursively denoted as the children of the given state. Clearly, the depth of the tree is ρ_{max}, and the breadth is of the order $o(3\rho_{max})$ since the order of the number of children of each state is 3. Because the number of possible sample paths could grow exponentially as Δt is reduced, a sampling of this population of sample paths is used to estimate the downtime distribution of the population. Accordingly, the two critical parameters in this estimation are the time interval Δt and the estimation convergence bound ε, where Δt controls the length of a sample path and ε determines the size of the sample needed for the estimation. We denote them as the *depth* and *breadth* parameters, respectively. Du et al. (2015) show that the estimator function $\hat{v}(\tau = \tau')$ derived using the SPR algorithm exhibits the statistical properties of unbiasedness, consistency, sufficiency, and is asymptotically normally distributed and asymptotically efficient. As the asymptotic normal distribution property of both $\hat{v}_S(\tau = \tau')$ and $\hat{\sigma}_S^2(\tau = \tau')$ is contingent on the sample size S, the size adequate for convergence was also determined. Note that the underlying birth-death process depends on the checkpointing strategy used. The SPR algorithm provides a general framework for sample path randomization for any birth-death process; accordingly, it can be adapted to any checkpointing strategy by using the transition probabilities of the corresponding birth-death process.

Algorithm 3.1 (The SPR Algorithm)

Input: parameters $\Delta t, \varepsilon$

Output: $\hat{v}(\tau = \tau')$

1. Select a minimum sample size S and compute downtime estimator $\hat{v}_S(\tau=\tau')$ and variance $\hat{\sigma}^2_S(\tau=\tau')$.

1.1 Generate S random sample paths:

For each $s=1,\ldots,S$, randomly generate a sample path d_s.

1.2 Derive $\hat{v}_S(\tau=\tau')$ and $\hat{\sigma}^2_S(\tau=\tau')$ for $\{d_1,d_2,\ldots,d_S\}$ using equations $\hat{v}_S(\tau=\tau')=\sum_{s=1}^{S} I_s(\tau')/S$, and $\hat{\sigma}^2_S=\sum_{s=1}^{S}(I_s(\tau')-\hat{v}_S(\tau=\tau'))^2/(S-1)$, where $I_s(\tau')$ is an index function for each path d_s: if the total downtime corresponding to a sample path $\tau(d_s)=\tau'$, $I_s(\tau')=1$; otherwise, $I_s(\tau')=0$.

2. Generate an additional sample path, setting $S\rightarrow S+1$ and derive $\hat{v}_{S+1}(\tau=\tau')$ and $\hat{\sigma}^2_{S+1}(\tau=\tau')$.

3. Evaluate stopping criterion: If $\max\limits_{\tau'}\{\frac{|\hat{v}_{S+1}(\tau=\tau')-\hat{v}_S(\tau=\tau')|}{\hat{v}_S(\tau=\tau')},\frac{|\hat{\sigma}^2_{S+1}(\tau=\tau')-\hat{\sigma}^2_S(\tau=\tau')|}{\hat{\sigma}^2_S(\tau=\tau')}\}>\varepsilon$, go to Step 2; otherwise, stop.

Now, consider the general M-VNF configuration. In this case, the system state is defined as a vector of the M single-VNF states: $\boldsymbol{IC}^{\rho}=[i^{\rho}_1,\ldots,i^{\rho}_M]$, where $\rho=0,\ldots,\rho_{max}$. The discrete-time Markov chain state transitions in a sample path of the M-VNF configuration is $\{\boldsymbol{IC}^0,\ldots,\boldsymbol{IC}^{\rho_{max}}\}$. In each sufficiently small time interval Δt, either a server from a VNF fails, a failed server in a VNF is repaired, or the system remains in the same state. So, the number of possible state transitions within each time interval is $2M+1$. We use this extended Markov chain within the SPR algorithm to generate sample paths and estimate $v(\tau)$ from them as the *integrated model* of the M-VNF configuration.

In the integrated model, while the depth of the tree of sample paths is still ρ_{max}, its breadth is of the order $o((2M+1)\rho_{max})$. Clearly, the size of the population of sample paths grows tremendously with M, and consequently would require significantly much larger sample sizes in obtaining statistically consistent estimates of $v(\tau)$. While imposing a significant computational burden as a result, the approach of directly using the integrated birth-death process of the M VNFs also suffers from problems of scalability, adaptability and algorithmic modularity as explained in the earlier section. Therefore, we develop a Markov chain decomposition strategy in the next section that yields fast and efficient estimation algorithms.

4. Convex Decomposition of Markov Chains

Our approach decomposes the Markov chain of the integrated model into M independent Markov chains, each pertaining to a single VNF. We first illustrate the decomposition idea using a 2-VNF case and subsequently generalize it to the M-VNF case.

4.1. Convex Decomposition of the 2-VNF Model

Consider a configuration consisting of two VNFs that are connected with one communication link. Let n_m, k_m, $m = 1, 2$, denote their respective primary and backup servers. For convenience, denote the current states of the two VNFs at the ρth time interval as $i_1^\rho = i$ and $i_2^\rho = j$, representing the number of failed VMs in the two VNFs, respectively. Since the communication link can only have two states, we denote the current state of the link as c, where $c = \{on\}$ or $\{off\}$. Let $\bar{c}$ denote the other state to which the link can transition to if it is currently in state c. Hence, if $c = \{on\}$, then $\bar{c} = \{off\}$, and vice versa. Accordingly, we denote the current state of the 2-VNF and 1-link configuration as a three-dimensional vector $\boldsymbol{IC}^\rho = [i, j, c]$. We assume that the system is available only when both the VNFs and the communication link are available concurrently. The transitions from any state $[i, j, c]$ in the underlying integrated Markov model are shown in Figure 2. For notational simplicity, we denote the transition probabilities as follows: the superscript 1, 2, and 3 on P index the two VNFs and the link, respectively, where the transition occurs; the two subscripts denote the state transition of the concerned VNF or link involved in the transition. Figure 3 presents a decomposition of this integrated chain into three independent chains, denoted as the i-chain, j-chain and c-chain, respectively. Note that the transition probabilities in this figure are accented with "$\sim$" to indicate the decomposed Markov chain for each independent VNF or link. The transition probabilities in the decomposed chains are derived from the original probabilities in the integrated model using the theory developed in the next section.

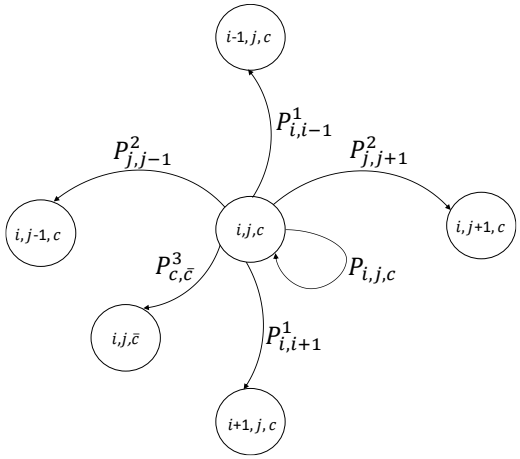

Figure 2 General State Transition in a 2-VNF and 1-link System

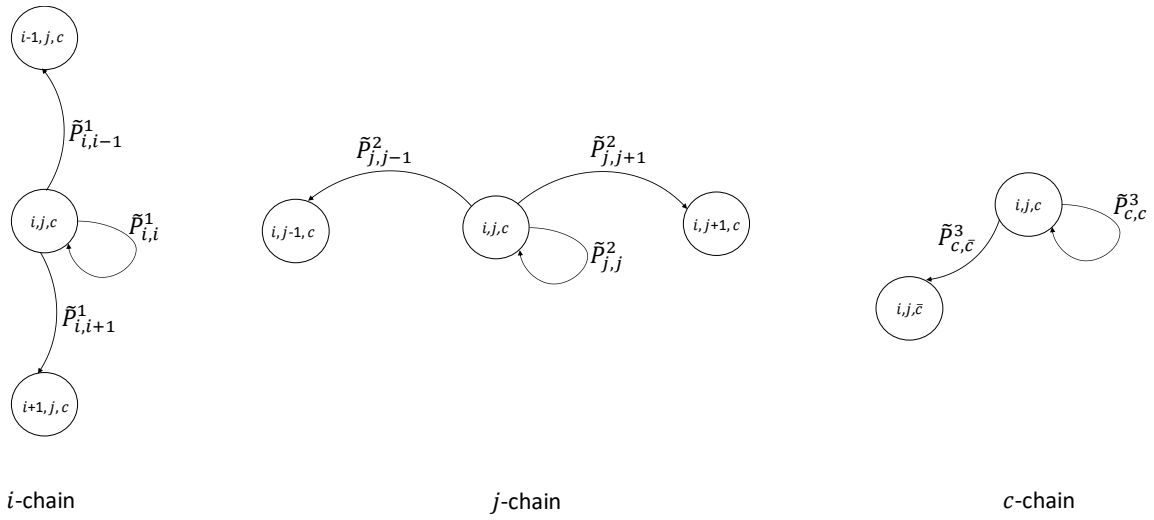

Figure 3 Decomposition of the 2-VNF and 1-link System

The idea behind this decomposition and the associated sample path randomization can be intuitively described as follows. Note that a transition in a time interval Δt can occur only in either one of the two VNFs or the single link. So, as a first step, we separate the transitions into the three chains as shown in Figure 3. Second, we select a set of convex non-negative multipliers δ_1, δ_2 and δ_3 such that $\delta_1 + \delta_2 + \delta_3 = 1$. Third, in any time interval Δt, we choose the i-chain, j-chain or the c-chain with probabilities δ_1, δ_2 and δ_3, respectively. Fourth, the transition in the time interval Δt is obtained from the chosen Markov chain, holding the states of the remaining chains unchanged. This process is repeated over the time slots $\rho = 1, 2, \ldots, \rho_{max}$. This process also yields a separate complete sample path for each chain over the entire time period. When we overlap these sample paths over all the time slots, each time slot maps to a specific chain chosen for transition in its corresponding Δt interval. Retrieving only the chosen chains and their transitions over all time slots and concatenating them yields a sample path of the integrated model.

The idea of decomposition into multiple independent Markov chains incorporates three distinct steps: selecting a chain for each time interval, generating a transition from the chosen chain in each interval, and finally, overlapping and concatenating the transitions over all the time intervals. This has three significant advantages. First, each VNF or link is treated independently, leading to modular and reusable sample path generation algorithms. Second, this approach does not depend on the number of VNFs or links involved; hence the algorithms are highly scalable. Third, the independence of the chains leads to any general

logical definition of availability of the overall system configuration; hence the algorithms are highly adaptable.

However, the decomposition strategy raises the following critical questions: (a) Does there exist a set of convex multipliers for any configuration such that the expected probabilities of transitions in the decomposed model are the same as the actual transition probabilities in the integrated model? (b) Given that the probabilities of transitioning out of a state are non-decreasing functions of Δt, is there an optimal set of multipliers and their corresponding optimal Δt's such that the computational time of the decomposition strategy is minimized? (c) Does the decomposition strategy yield downtime distributions that are statistically consistent with those of the integrated model? We address these questions in the following discussion.

4.2. Convex Decomposition of the *M*-VNF Model

Consider a system consisting of M VNFs and L links. The system state at step ρ transition is a path-dependent $(M+L)$-dimensional vector $\boldsymbol{IC}^\rho = [i_1^\rho, \ldots, i_M^\rho, c_1^\rho, \ldots, c_L^\rho]$. To simplify notations, we omit the time interval index ρ. Denote $i_m = 0, \ldots, n_m + k_m, m = 1, \ldots, M$, as the current state associated with VNF m, and $c_l = \{on, off\}$, $l = 1, \ldots, L$, the current state associated with link l. Because at most one event can occur in a sufficiently small time interval, in the case that VNF m fails one server, we denote the transition probability as $P^m_{i_m, i_m+1}$. Similarly, in the case that a server is repaired in VNF m, we denote the transition probability as $P^m_{i_m, i_m-1}$. In the event of a link failure or repair, we denote the transition probability as $P^l_{c_l, \bar{c}_l}$, where $\bar{c}_l$ is the state that a link transits out of state c_l. Further denote $P_{\boldsymbol{IC}}$ as the probability that the system remains in the current state.

Note that as Δt increases, there is increasing probability that a failure or repair event would occur. Therefore, $P^m_{i_m, i_m+1}$, $P^m_{i_m, i_m-1}$ and $P^l_{c_l, \bar{c}_l}$ are non-decreasing functions of Δt, and $P_{\boldsymbol{IC}}$ is a non-increasing function of Δt. To simplify our notations and without causing confusion, we suppress Δt in the general state probability expressions except in the balance equations (1)-(3) and in the boundary conditions (4)-(5) where we would like to emphasize its dependence on Δt. The following balance equation holds at any time interval $\rho = 1, \ldots, \rho_{max}$:

$$P_{\boldsymbol{IC}}(\Delta t) + \sum_{m=1}^{M} \{P^m_{i_m, i_m-1}(\Delta t) + P^m_{i_m, i_m+1}(\Delta t)\} + \sum_{l=1}^{L} P^l_{c_l, \bar{c}_l}(\Delta t) = 1 \tag{1}$$

Now, consider $M+L$ decomposed Markov chains. With probabilities $\delta_m, m=1,\ldots,M$, a node transition occurs; similarly, with probabilities $\delta_l, l=1,\ldots,L$, a link transition occurs. Moreover, $\sum_{m=1}^{M}\delta_m+\sum_{l=1}^{L}\delta_l=1$. Let $\tilde{P}^m_{i_m,i_m-1}$, $\tilde{P}^m_{i_m,i_m}$ and $\tilde{P}^m_{i_m,i_m+1}$, $m=1,\ldots,M$, denote the transition probabilities in the decomposed Markov chain of VNF m, and $\tilde{P}^l_{c_l,\bar{c}_l}$, where $c_l=\{on,off\}$ and $l=1,\ldots,L$, denote the transition probabilities in the decomposed Markov chain of communication link l. Again, the following balance equations in the decomposed system holds:

$$\begin{cases}\tilde{P}^m_{i_m,i_m-1}(\Delta t)+\tilde{P}^m_{i_m,i_m}(\Delta t)+\tilde{P}^m_{i_m,i_m+1}(\Delta t)=1\\ \tilde{P}^l_{c_l,c_l}(\Delta t)+\tilde{P}^l_{c_l,\bar{c}_l}(\Delta t)=1\end{cases}\tag{2}$$

4.2.1. Multiplier Existence Theorem. The following theorem establishes the existence of a feasible set of convex multipliers for decomposing the integrated model of any configuration that consists of M VNFs and L links.

THEOREM 1 **(Existence of Convex Decomposition Multipliers)**. *At each step of the state transition, there exists a set of convex multipliers δ_m's and δ_l's such that $0\le\delta_m\le 1, 0\le\delta_l\le 1, \sum_{m=1}^{M}\delta_m+\sum_{l=1}^{L}\delta_l=1$, to decompose the Markov chain of the integrated system into $M+L$ independent Markov chains of single VNFs and links, where the transition probabilities in the integrated model equal their expected probabilities in the decomposed model:*

$$\begin{cases}P_{\boldsymbol{IC}}(\Delta t)=\sum_{m=1}^{M}\delta_m\tilde{P}^m_{i_m,i_m}(\Delta t)+\sum_{l=1}^{L}\delta_l\tilde{P}^l_{c_l,c_l}(\Delta t)\\ P^m_{i_m,i_m+1}(\Delta t)=\delta_m\tilde{P}^m_{i_m,i_m+1}(\Delta t)\\ P^m_{i_m,i_m-1}(\Delta t)=\delta_m\tilde{P}^m_{i_m,i_m-1}(\Delta t)\\ P^l_{c_l,\bar{c}_l}(\Delta t)=\delta_l\tilde{P}^l_{c_l,\bar{c}_l}(\Delta t), where\, c_l=\{on,off\}.\end{cases}\tag{3}$$

Proof. Given the correspondence in (3), we first show the balance equation (1) in the integrated model's Markov system holds. We then show such feasible set of multipliers exist. First, we have $P_{\boldsymbol{IC}}+\sum_{m=1}^{M}\{P^m_{i_m,i_m-1}+P^m_{i_m,i_m+1}\}+\sum_{l=1}^{L}P^l_{c_l,\bar{c}_l}=P_{\boldsymbol{IC}}+\sum_{m=1}^{M}\delta_m\{\tilde{P}^m_{i_m,i_m-1}+\tilde{P}^m_{i_m,i_m+1}\}+\sum_{l=1}^{L}\delta_l\tilde{P}^l_{c_l,\bar{c}_l}=P_{\boldsymbol{IC}}+\sum_{m=1}^{M}\delta_m\{1-\tilde{P}^m_{i_m,i_m}\}+\sum_{l=1}^{L}\delta_l\{1-\tilde{P}^l_{c_l,c_l}\}=P_{\boldsymbol{IC}}+\sum_{m=1}^{M}\delta_m+\sum_{l=1}^{L}\delta_l-[\sum_{m=1}^{M}\delta_m\tilde{P}^m_{i_m,i_m}+\sum_{l=1}^{L}\delta_l\tilde{P}^l_{c_l,c_l}]=1$. So (1) holds. Next, we show that feasible δ_m's and δ_l's exist. Because $\tilde{P}^m_{i_m,i_m-1}=P^m_{i_m,i_m-1}/\delta_m$, $\tilde{P}^m_{i_m,i_m+1}=P^m_{i_m,i_m+1}/\delta_m$, feasibility conditions require that: $P^m_{i_m,i_m-1}+P^m_{i_m,i_m+1}\le\delta_m$, $m=1,\ldots,M$. Similarly, because $\tilde{P}^l_{c_l,\bar{c}_l}=P^l_{c_l,\bar{c}_l}/\delta_l$, feasibility condition requires that: $P^l_{c_l,\bar{c}_l}\le\delta_l$, $l=1,\ldots,L$. Therefore, $P^M_{i_M,i_M-1}+P^M_{i_M,i_M+1}\le\delta_M=1-\sum_{m=1}^{M-1}\delta_m-\sum_{l=1}^{L}\delta_l$. So $\delta_1\le 1-\sum_{m=2}^{M-1}\delta_m-$

$\sum_{l=1}^{L}\delta_l - \{P^M_{i_M,i_M-1} + P^M_{i_M,i_M+1}\}$. We next show δ_1 exists. Other δ_m's and δ_l's can be shown in a similar way. Because $-\delta_m \le -\{P^m_{i_m,i_m-1} + P^m_{i_m,i_m+1}\}$, $m = 2,\ldots,M-1$, and $-\delta_l \le -P^l_{c_l,\bar{c}_l}$, $l = 1,\ldots,L$, we have $\delta_1 \le 1 - \sum_{m=2}^{M-1}\delta_m - \sum_{l=1}^{L}\delta_l - \{P^M_{i_M,i_M-1} + P^M_{i_M,i_M+1}\} \le 1 - \sum_{m=2}^{M}\{P^m_{i_m,i_m-1} + P^m_{i_m,i_m+1}\} - \sum_{l=1}^{L}P^l_{c_l,\bar{c}_l} = P_{\boldsymbol{IC}} + P^1_{i_1,i_1-1} + P^1_{i_1,i_1+1}$. Therefore, this equation together with $P^1_{i_1,i_1-1} + P^1_{i_1,i_1+1} \le \delta_1$ define the feasible range of δ_1. By similar reasoning, there exists individual feasible δ_m and δ_l such that:

$$P^m_{i_m,i_m-1}(\Delta t) + P^m_{i_m,i_m+1}(\Delta t) \le \delta_m \le P_{\boldsymbol{IC}}(\Delta t) + P^m_{i_m,i_m-1}(\Delta t) + P^m_{i_m,i_m+1}(\Delta t), m = 1,\ldots,M, \tag{4}$$

and

$$P^l_{c_l,\bar{c}_l}(\Delta t) \le \delta_l \le P_{\boldsymbol{IC}}(\Delta t) + P^l_{c_l,\bar{c}_l}(\Delta t), l = 1,\ldots,L. \tag{5}$$

Summing over all δ_m's and δ_l's, we have $1 - P_{\boldsymbol{IC}} = \sum_{m=1}^{M}\{P^m_{i_m,i_m-1} + P^m_{i_m,i_m+1}\} + \sum_{l=1}^{L}P^l_{c_l,\bar{c}_l} \le \sum_{m=1}^{M}\delta_m + \sum_{l=1}^{L}\delta_l \le (M+L)P_{\boldsymbol{IC}} + \sum_{m=1}^{M}\{P^m_{i_m,i_m-1} + P^m_{i_m,i_m+1}\} + \sum_{l=1}^{L}P^l_{c_l,\bar{c}_l} = (M+L)P_{\boldsymbol{IC}} + 1 - P_{\boldsymbol{IC}} = (M+L-1)P_{\boldsymbol{IC}} + 1$. The aggregated lower bound $1 - P_{\boldsymbol{IC}} \le 1$ and the aggregated upper bound $(M+L-1)P_{\boldsymbol{IC}} + 1 \ge 1$. So there are feasible δ_m's and δ_l's such that $\sum_{m=1}^{M}\delta_m + \sum_{l=1}^{L}\delta_l = 1$. □

Theorem 1 shows equivalence between the transition probabilities in the integrated model and their expectations in the decomposed model. Inequalities (4) and (5) suggest that there exist a continuous range of feasible values of the convex multipliers to decompose the integrated model. Therefore, a natural question is whether there exists an optimal set of convex multipliers to achieve computational efficiency in the decomposition algorithm. We answer this question in the next subsection.

4.2.2. Optimal Convex Decomposition. As discussed, the computational performance of the algorithm depends on both the breadth (ε) and depth (ρ_{max}) parameters. The breadth parameter ε affects the total number of sample paths needed so that the distributional properties of the sample reasonably represent the population. The smaller the ε, the larger the required sample size S. Because the depth parameter $\rho_{max} = \lceil T/\Delta t \rceil$, larger Δt is computationally desirable. By choosing the maximum allowable time interval in each

step of the state transition, we can reduce the total number of intervals for a sample path. This decision problem can be formally stated as:

$$
\begin{aligned}
\Delta t_{\max} \quad &= \max_{\delta_m, \delta_l, \Delta t} \Delta t \\
\text{s.t.} \quad & P_{\boldsymbol{IC}}(\Delta t) \geq 0 \\
& \sum_{m=1}^{M} \delta_m + \sum_{l=1}^{L} \delta_l = 1 \\
& P^m_{i_m, i_m - 1}(\Delta t) + P^m_{i_m, i_m + 1}(\Delta t) \leq \delta_m \leq P_{\boldsymbol{IC}}(\Delta t) + P^m_{i_m, i_m - 1}(\Delta t) + P^m_{i_m, i_m + 1}(\Delta t), m = 1, \ldots, M \\
& P^l_{c_l, \bar{c}_l}(\Delta t) \leq \delta_l \leq P_{\boldsymbol{IC}}(\Delta t) + P^l_{c_l, \bar{c}_l}(\Delta t), l = 1, \ldots, L \\
& Eq.(3)
\end{aligned}
\tag{DP}
$$

Our objective is to find the maximum allowable Δt in each step of the state transitions for the decomposition approach to work. The first constraint is simply the non-negativity constraint for probability. The second constraint is the convex combination condition for the decomposition multipliers. The third and fourth sets of constraints (from Equations 4 and 5) characterize the feasible ranges of (δ_m, δ_l), for $m = 1, \ldots, M$ and $l = 1, \ldots, L$. The last constraint ensures the state transition probability correspondences between the decomposed and integrated model as established in Equation (3).

The value of Δt would affect the feasible ranges of the multipliers. Since the larger the Δt, the smaller the chance a VNF or a link component will stay in its current state, $\tilde{P}^m_{i_m, i_m}$ and $\tilde{P}^l_{c_l, c_l}$ are non-increasing functions of Δt. As a result, $P_{\boldsymbol{IC}}$ is a non-increasing function of Δt. In any step of transition, the feasibility condition $P_{\boldsymbol{IC}} \geq 0$ should hold and $P_{\boldsymbol{IC}} = 0$ defines an upper bound of Δt to ensure a well-defined Markov chain transition.

Because the overall optimization involves $(M + L + 1)$ decision variables and several sets of conditions, it may be computationally costly to find an optimal decomposition. However, we can first solve the relaxed optimization problem based on the integrated model:

$$
\begin{aligned}
\Delta t_{\max} \quad &= \max_{\Delta t} \Delta t \\
\text{s.t.} \quad & P_{\boldsymbol{IC}}(\Delta t) \geq 0 \\
& P_{\boldsymbol{IC}}(\Delta t) + \sum_{m=1}^{M} \{P^m_{i_m, i_m - 1}(\Delta t) + P^m_{i_m, i_m + 1}(\Delta t)\} + \sum_{l=1}^{L} P^l_{c_l, \bar{c}_l}(\Delta t) = 1
\end{aligned}
\tag{IP}
$$

Different from (DP) in which the decomposition constraints should be satisfied and the probability relationship between the integrated and decomposed models is maintained, the problem (IP) only considers the integrated model. The second constraint (from Equation 1) ensures that $P_{\boldsymbol{IC}}$ satisfies the integrated system balance equation. Because the transition-out probabilities $P^m_{i_m,i_m-1}$, $P^m_{i_m,i_m+1}$, and $P^l_{c_l,\bar{c}_l}$ are non-decreasing functions of Δt, the maximum Δt is obtained when $P_{\boldsymbol{IC}}=0$. The following Theorem shows that the solution Δt_{max} from the relaxed optimization problem (IP) uniquely determines the set of optimal convex decomposition multipliers δ_m's and δ_l's, which constitute the optimal solution to (DP).

THEOREM 2 **(Optimal Decomposition)**. *The optimal solution Δt_{max} is obtained from the integrated model (IP) by solving $P_{\boldsymbol{IC}}(\Delta t)=0$, which uniquely defines the set of optimal convex decomposition multipliers in (DP):*

$$\delta^*_m = P^m_{i_m,i_m-1}(\Delta t_{max}) + P^m_{i_m,i_m+1}(\Delta t_{max}), m=1,\ldots,M, \tag{6}$$

$$\delta^*_l = P^l_{c_l,\bar{c}_l}(\Delta t_{max}), l=1,\ldots,L. \tag{7}$$

In addition, (Δt_{max}, δ^*_m,δ^*_l) constitutes the optimal solution to the decomposed model (DP).

Proof. Consider the relaxed optimization problem (IP). By the balance Equation (1), $P_{\boldsymbol{IC}} = 1 - \sum_{m=1}^{M}\{P^m_{i_m,i_m-1} + P^m_{i_m,i_m+1}\} - \sum_{l=1}^{L}\{P_{c_l,\bar{c}_l}\}$. Because the VNFs and links' transition-out probabilities are non-decreasing in Δt, $P_{\boldsymbol{IC}}$ is non-increasing in Δt. The maximum Δt is determined by solving $P_{\boldsymbol{IC}} = 0$. From (4) and (5) we see that, when $P_{\boldsymbol{IC}} = 0$, each decomposition multiplier's upper bound equals the lower bound. This is exactly the maximum allowable Δt beyond which no feasible decomposition strategy exists. So the optimal set of decomposition multipliers are set at these boundary values, as shown in Equations (6) and (7). □

Theorem 1 suggests that, corresponding to every Δt for a well-defined Markov system, there exists a feasible set of decomposition multipliers to decompose the integrated system. Theorem 2 further suggests that the optimal set of decomposition multipliers is uniquely defined by the maximum allowable Δt of the integrated Markov chain. It implies that the same Δt_{max} optimizes both the integrated model and the decomposed model's computational performance by minimizing the total number of steps required to complete sampling a sample path.

So far, we have expressed the transition probabilities in general terms. Our decomposition strategy works for any general VM server failure and repair distribution. In the following, we give the explicit expressions for these transition probabilities. Specifically, let λ_m and μ_m, where $m = 1, \ldots, M$, be the failure and repair rates of VNF m in the integrated model. We have:

$$P^m_{i_m, i_m - 1} = i_m \mu_m \Delta t, \tag{8}$$

$$P^m_{i_m, i_m + 1} = (n_m + k_m - i_m)\lambda_m \Delta t. \tag{9}$$

Assume link l's failure and repair rates are λ_l and μ_l, for $l = 1, \ldots, L$. Then the link transition probability is:

$$P^l_{c_l, \bar{c}_l} = r_{c_l}\Delta t = \begin{cases} \lambda_l \Delta t & if \quad c_l = on \\ \mu_l \Delta t & if \quad c_l = off \end{cases} \tag{10}$$

Based on Theorem 2 and the specifications in Equations (8)-(10), the following Corollary determines the maximum allowable Δt for the system-wide convex decomposition.

COROLLARY 1 **(Computationally Efficient Decomposition Strategy)**. *The most computationally efficient decomposition strategy is to choose Δt_{max} and the decomposition multipliers at each step of transition as follows:*

$$\Delta t_{max} = \frac{1}{\sum_{m=1}^{M}[(n_m + k_m)\lambda_m + i_m(\mu_m - \lambda_m)] + \sum_{l=1}^{L} r_{c_l}} \tag{11}$$

$$\delta^*_m = [(n_m + k_m)\lambda_m + i_m(\mu_m - \lambda_m)]\Delta t_{max}, m = 1, \ldots, M, \tag{12}$$

$$\delta^*_l = r_{c_l}\Delta t_{max}, l = 1, \ldots, L. \tag{13}$$

Proof. $P_{\boldsymbol{IC}} = 1 - \sum_{m=1}^{M}\{P^m_{i_m, i_m - 1} + P^m_{i_m, i_m + 1}\} - \sum_{l=1}^{L} P^l_{c_l, \bar{c}_l} = 1 - \sum_{m=1}^{M} \Delta t[(n_m + k_m)\lambda_m + i_m(\mu_m - \lambda_m)] - \sum_{l=1}^{L} r_{c_l}\Delta t$. As Δt increases, $P_{\boldsymbol{IC}}$ decreases. The largest Δt that ensures the non-negative probability is determined by solving $P_{\boldsymbol{IC}} = 0$, which yields (11). Because $P_{\boldsymbol{IC}} = 0$, the feasible ranges defined by (4) and (5) shrink to point solutions at the boundary values. Substituting (11) into the lower bound expression we have (12) and (13). □

4.2.3. Statistical Equivalence Property. Denote $\tilde{\lambda}_m$, $\tilde{\mu}_m$ and $\tilde{r}_{c_l}$ as the failure/repair rates in the decomposed model. Equations (8)-(10) state the transition-out probabilities in the integrated model. Similarly, we have $\tilde{P}^m_{i_m,i_m-1} = i_m\tilde{\mu}_m\Delta t$, $\tilde{P}^m_{i_m,i_m+1} = (n_m + k_m - i_m)\tilde{\lambda}_m\Delta t$, $\tilde{P}^l_{c_l,\bar{c}_l} = \tilde{r}_{c_l}\Delta t$ in the decomposed model. The following Theorem establishes the statistical equivalence between the integrated and decomposed approaches.

THEOREM 3 **(Statistical Equivalence Property)**. *The decomposed model produces statistically equivalent downtime distribution as the integrated model if the failure and repair rates in the two models have the following relationships for* $m = 1, \ldots, M$ *and* $l = 1, \ldots, L$*:*

$$\begin{cases} \tilde{\lambda}_m = \lambda_m/\delta_m \\ \tilde{\mu}_m = \mu_m/\delta_m \\ \tilde{r}_{c_l} = r_{c_l}/\delta_l \; where \; c_l = \{on, off\}. \end{cases} \tag{14}$$

Proof. First, $P^m_{i_m,i_m-1} = i_m\mu_m\Delta t = \delta_m i_m(\mu_m/\delta_m)\Delta t = \delta_m i_m\tilde{\mu}_m\Delta t = \delta_m\tilde{P}^m_{i_m,i_m-1}$. Second, $P^m_{i_m,i_m+1} = (n_m + k_m - i_m)\lambda_m\Delta t = \delta_m(n_m + k_m - i_m)(\lambda_m/\delta_m)\Delta t = \delta_m(n_m + k_m - i_m)\tilde{\lambda}_m\Delta t = \delta_m\tilde{P}^m_{i_m,i_m+1}$. Third, $P^l_{c_l,\bar{c}_l} = r_{c_l}\Delta t = \delta_l(r_{c_l}/\delta_l)\Delta t = \delta_l\tilde{r}_{c_l}\Delta t = \delta_l\tilde{P}^l_{c_l,\bar{c}_l}$. Finally, $\sum_{m=1}^M \delta_m\tilde{P}^m_{i_m,i_m} + \sum_{l=1}^L \delta_l\tilde{P}^l_{c_l,c_l} = \sum_{m=1}^M \delta_m[1 - \tilde{P}^m_{i_m,i_m-1} - \tilde{P}^m_{i_m,i_m+1}] + \sum_{l=1}^L \delta_l[1 - \tilde{P}^l_{c_l,\bar{c}_l}] = \sum_{m=1}^M \delta_m[1 - P^m_{i_m,i_m-1}/\delta_m - P^m_{i_m,i_m+1}/\delta_m] + \sum_{l=1}^L \delta_l[1 - P^l_{c_l,\bar{c}_l}/\delta_l] = 1 - \sum_{m=1}^M[P^m_{i_m,i_m-1} + P^m_{i_m,i_m+1}] - \sum_{l=1}^L \delta_l P^l_{c_l,\bar{c}_l} = P_{\boldsymbol{IC}}$. Therefore, on expectation the decomposed Markov chain produces the same transition probability as the integrated Markov chain, as Equation (3) shows. The downtime distribution produced by the integrated model and the decomposed model is statistically equivalent. □

5. Algorithmic Implementation Framework

Using the integrated model as a benchmark, we develop several variations of the SPR algorithm under convex decomposition for estimating the transient downtime distributions of system configurations with M VNFs and L communication links. The key difference in the different implementation strategies is in the use of the Δt parameter in the SPR algorithm. The first implementation uses a fixed Δt model at each transition step of the sample path generation, while the second uses a variable Δt model. In the fixed Δt model, a fixed step size Δt_{fix} is used to define the time interval, so the total number of intervals is $\rho_{max} = \lceil \frac{T}{\Delta t_{fix}} \rceil$. In the variable Δt model, the step size of each state transition is determined

by the optimal solution to the problem (IP), given as Δt_{max}. Thus ρ_{max} is state dependent and the total number of time intervals would be different for each random sample path.

In the decomposed model, we need to compute the decomposition multipliers δ_m's and δ_l's. If we implement a variable Δt decomposition, δ_m^*'s, δ_l^*'s and Δt_{max} are determined as in Corollary 1. If we implement a fixed Δt decomposition, then substituting Δt into Equations (12) and (13) we have feasible decomposition multipliers δ_m's and δ_l's determined by their respective lower bounds in inequalities (4) and (5). Because the computation of the decomposition multipliers takes some CPU time, we further propose a simple heuristic that does not require such computations. Different from other model implementations in which at most one event can occur in each step of the state transition, the heuristic algorithm allows each network component (a VNF or link) to operate simultaneously and independently. So multiple events can occur in some time intervals in the heuristic model. We must carefully calibrate Δt in this case to ensure the occurrence of multiple events is controlled in an acceptable range, so that the heuristic would still yield consistent downtime distribution as in the integrated and decomposed models.

The transition probabilities in each model are determined by the underlying VM failure/repair distributions and the time interval (either variable or fixed) in each transition step. Let $t \in [0, T]$ denote time elapsed in the service window. The Exponential failure assumes a constant hazard rate function $\lambda_m(t) = \lambda_m, m = 1, \ldots, M + L$. We denote the $(M + L)$-dimensional vector as $\boldsymbol{\lambda_E} = [\lambda_1, \ldots, \lambda_{M+L}]'$. Weibull distribution has a time-varying hazard rate function $\boldsymbol{\lambda_W}(t) = [\lambda_1(t), \ldots, \lambda_{M+L}(t)]'$ with $\lambda_m(t) = \alpha_m \beta(\alpha_m t)^{\beta-1}, m = 1, \ldots, M + L$, where $\alpha_m > 0$ is a scale parameter and $\beta > 0$ is a shape parameter. For $\beta = 1$, the Weibull distribution yields constant failure rate α_m, which gives the Exponential distribution. For $\beta > 1$, the Weibull distribution models an increasing failure rate such as servers "aging" over time. For $\beta < 1$, the Weibull distribution models decreasing failure rate, which represents "infant mortality" where defective components are rectified over time. The Erlang distribution is also defined by two parameters: the scale parameter $e_m > 0$ and the integer shape parameter γ. Its hazard rate function $\boldsymbol{\lambda_R}(t) = [\lambda_1(t), \ldots, \lambda_{M+L}(t)]'$, where $\lambda_m(t) = \frac{e_m (e_m t)^{\gamma-1}}{(\gamma-1)! S_{\gamma-1}(t)}, m = 1, \ldots, M + L$, and $S_{\gamma-1}(t) = 1 + e_m t + \frac{(e_m t)^2}{2!} + \ldots + \frac{(e_m t)^{r-1}}{(\gamma-1)!}$ is the partial sum to γ terms of the Exponential series expansion of $e^{e_m t}$. When γ=1, the Erlang distribution yields constant failure rate e_m, which is the Exponential distribution. When $\gamma > 1$, it represents the increasing failure rate.

Assume the real starting time for interval ρ is t_ρ. Since the interval is small enough, we assume both the failure rate and repair rate remain constant in this interval. We thus can compute the failure rate $\lambda_m(t_\rho), m = 1, \ldots, M + L$, for the VNFs and links in time interval ρ based on the distributional assumption. By adapting Equations (8), (9) and (10) we can construct the transition probability in the ρth time interval.

In the integrated model implementation, the transition probability matrix $P(\rho)$ is a single matrix with VNF and link probabilities defined as: $P^m_{i_m,i_m-1}(\rho) = i^\rho_m \mu_m \Delta t$, $P^m_{i_m,i_m+1}(\rho) = (n_m + k_m - i^\rho_m)\lambda_m(t_\rho)\Delta t$, for $m = 1, \ldots, M$; $P^l_{c_l,\bar{c}_l}(\rho) = \lambda_l(t_\rho)\Delta t$, if $c_l = \{on\}$ and $P^l_{c_l,\bar{c}_l}(\rho) = \mu_l \Delta t$, if $c_l = \{off\}$, for $l = 1, \ldots, L$; and $P_{\boldsymbol{IC}}(\rho) = 1 - \sum_{m=1}^{M}\{P^m_{i_m,i_m-1}(\rho) + P^m_{i_m,i_m+1}(\rho)\} - \sum_{l=1}^{L} P^l_{c_l,\bar{c}_l}(\rho)$. The integrated model with M VNFs and L links has $(2M + L + 1)$ states.

In the decomposed model implementation, define $\tilde{\mathbf{P}}(\rho) = [\tilde{\mathbf{P}}^1(\rho), \ldots, \tilde{\mathbf{P}}^{M+L}(\rho)]$. The transition probability matrix $\tilde{\mathbf{P}}(\rho)$ consists of $M + L$ independent state transition matrices. The component probabilities $\tilde{\mathbf{P}}^m(\rho)$ are defined as: $\tilde{P}^m_{i_m,i_m-1}(\rho) = i^\rho_m \tilde{\mu}_m \Delta t$, $\tilde{P}^m_{i_m,i_m+1}(\rho) = (n_m + k_m - i^\rho_m)\tilde{\lambda}_m(t_\rho)\Delta t$ and $\tilde{P}^m_{i_m,i_m}(\rho) = 1 - \tilde{P}^m_{i_m,i_m-1}(\rho) - \tilde{P}^m_{i_m,i_m+1}(\rho)$ for $m = 1, \ldots, M$. The component probabilities $\tilde{\mathbf{P}}^l(\rho)$ are defined as: $\tilde{P}^l_{c_l,\bar{c}_l}(\rho) = \lambda_l(t_\rho)\Delta t$, if $c_l = \{on\}$ and $\tilde{P}^l_{c_l,\bar{c}_l}(\rho) = \mu_l \Delta t$, if $c_l = \{off\}$, for $l = 1, \ldots, L$. Because only one state transition vector $\tilde{\mathbf{P}}^m(\rho)$ or $\tilde{\mathbf{P}}^l(\rho)$ is randomly chosen in each step, dimensionality is dramatically reduced in the decomposed model (3 for a VNF and 2 for a link).

In the time interval ρ, define binary variables $A_{m\rho} = 1, m = 1, \ldots, M$, if VNF m is available, and $B_{l\rho} = 1, l = 1, \ldots, L$, if link l is available. Any logical combination of VNF availability on $A_{m\rho}, m = 1, \ldots, M$, and link availability on $B_{l\rho}, l = 1, \ldots, L$, can be defined as overall system availability. We denote this as the system availability logic $\mathcal{L}$. Without loss of generality, we have assumed that all VNFs and links should be concurrently available for the overall system to be available in our experiments, and VNF m is available in a time interval ρ only when no more than k_mVMs fail simultaneously in that time interval. Note that any definition of availability can be used in our proposed algorithm. The network availability logic does not affect the performance of the decomposition algorithm.

In the following, we present an integrated framework for estimating the overall transient downtime of a M-VNF, L-link system. This framework incorporates three specific user implementation choices: *Distribution_type* (Exponential, Weibull, Erlang), *Model_type* (Integrated, Decomposed, Heuristic), *Step_type* (Fixed Δt, Variable Δt). Algorithm 5.1 shows the different algorithmic implementations collectively presented as a framework

based on the system input parameters and user choices. Besides these, the user would also specify the system availability logic, the initial minimum number of sample paths to be generated and the convergence bound.

Algorithm 5.1 (Framework for estimating transient downtime of M-VNF, L-link systems)

Input:

1. *System parameters*: M, L, n_m, k_m, for $m = 1, \ldots, M$

   Repair process rates: μ_m, for $m = 1, \ldots, M+L$

   For Exponential failures: λ_m, for $m = 1, \ldots, M+L$

   For Weibull failures: scale parameters α_m, for $m = 1, \ldots, M+L$; shape parameter β

   For Erlang failures: scale parameters e_m, for $m = 1, \ldots, M+L$; shape parameter γ

2. *User choices*: System availability logic $\mathcal{L}$; Step_type, Model_type, Distribution_type, Δt_{fix}, initial number of sample paths S, convergence bound ε

Output: downtime distribution $\hat{v}(\tau = \tau')$

Module: **MAIN**

Begin

For $s = 1, \ldots, S$ **Do**

SamplePath(s);

// will return I_s

Calculate the mean of the overall system downtime $\hat{v}_S(\tau = \tau') = \sum_{s=1}^{S} I_s(\tau')/S$ for all τ';

Calculate the variance of the overall system downtime $\hat{\sigma}_S^2(\tau = \tau') = \sum_{s=1}^{S}(I_s(\tau') - \hat{v}_S(\tau = \tau'))^2/(S-1)$ for all τ';

$\varphi = 0$;

Repeat

SamplePath(S+1);

Calculate $\hat{v}_{S+1}$ and $\hat{\sigma}_{S+1}^2$;

If $\max\limits_{\tau'}\{\frac{|\hat{v}_{S+1}(\tau=\tau') - \hat{v}_S(\tau=\tau')|}{\hat{v}_S(\tau=\tau')}, \frac{|\hat{\sigma}_{S+1}^2(\tau=\tau') - \hat{\sigma}_S^2(\tau=\tau')|}{\hat{\sigma}_S^2(\tau=\tau')}\} \leq \varepsilon$

$\varphi = 1$;

Else

$S \to S + 1$;

Until φ=1;

End

Module: **SamplePath**(s)

Begin

$\rho = 0$;

$t = 0$;

Initialize i_m^0, for $m = 1, \ldots, M$ and c_l^0 for $l = 1, \ldots, L$;

Initialize $\boldsymbol{\lambda_E}$, $\boldsymbol{\lambda_W}(0)$, or $\boldsymbol{\lambda_R}(0)$ based on the Distribution_type;

While $t < T$ *//generate one random sample path*

If Model_type = integrated //select model type

If Step_type = variable //select state transition type

Calculate $\Delta t = \Delta t_{max}$;

Else

$\Delta t = \Delta t_{fix}$;

Obtain the transition probability matrix $\mathbf{P}(\rho)$ based on Distribution_type;

Starting from $\boldsymbol{IC}^{\rho}$, randomly generate a transition using $\mathbf{P}(\rho)$;

Let $\boldsymbol{IC}^{\rho+1}$ denote the resulting state vector;

$\rho \to \rho + 1$;

$t \to t + \Delta t$;

Determine VNF and link availability related to a state transition: $A_{m\rho}$ and $B_{l\rho}$;

Determine system availability using $\mathcal{L}$: $I_\rho = \mathcal{L}(A_{m\rho}, B_{l\rho}) = 1$ if the overall system is available; otherwise, $I_\rho = 0$;

Calculate the system uptime in the current time interval: $I_\rho \Delta t$;

If Model_type = decomposed

If Step_type = variable

Calculate $\Delta t = \Delta t_{max}$;

Else

$\Delta t = \Delta t_{fix}$;

Obtain the transition probability matrix $\tilde{\mathbf{P}}(\rho)$ based on Distribution_type;

Derive the decomposition multipliers δ_m, $m = 1, \ldots, M$, and δ_l, $l = 1, \ldots, L$;

Using the decomposition multiplier, randomly select a VNF or link;

Let m or l denote the selected VNF or link, respectively;

Starting from i_m^ρ or c_l^ρ, randomly generate a transition using $\tilde{\mathbf{P}}^m(\rho)$;

Let $i_m^{\rho+1}$ or $c_l^{\rho+1}$ denote the resulting state;

$\rho \to \rho + 1$;

$t \to t + \Delta t$;

Determine VNF and link availability related to a state transition: $A_{m\rho}$ and $B_{l\rho}$;

Determine system availability using $\mathcal{L}$: $I_\rho = \mathcal{L}(A_{m\rho}, B_{l\rho}) = 1$ if the overall system is available; otherwise, $I_\rho = 0$;

Calculate the system uptime in the current time interval: $I_\rho \Delta t$;

Else // *heuristic*

If Step_type = variable

Calculate $\Delta t = \Delta t_{max}$;

Else

$\Delta t = \Delta t_{fix}$;

Obtain the transition probability matrix $\tilde{\mathbf{P}}(\rho)$ based on Distribution_type;

Starting from i_m^ρ and c_l^ρ, independently and randomly generate a transition using $\tilde{\mathbf{P}}^m(\rho)$ for $m = 1, \ldots, M$ and $l = 1, \ldots, L$;

Let $i_m^{\rho+1}$, $m = 1, \ldots, M$,and $c_l^{\rho+1}$, $l = 1, \ldots, L$ denote the resulting state;

$\rho \to \rho + 1$;

$t \to t + \Delta t$;

Determine VNF and link availability related to a state transition: $A_{m\rho}$ and $B_{l\rho}$;

Determine system availability using $\mathcal{L}$: $I_\rho = \mathcal{L}(A_{m\rho}, B_{l\rho}) = 1$ if the overall system is available; otherwise, $I_\rho = 0$;

Calculate the system uptime in the current time interval: $I_\rho \Delta t$;

End While

Calculate the percentage of system downtime for a sample path $\tau_s = 1 - \sum_{t<T} I_\rho \Delta t / T$;

// *map the actual downtime of each system sample path to discretized τ's as follows:*

Let τ' denote the smallest value for which $\tau_s \le \tau'$ to hold;

$I_s(\tau') = 1$;

$I_s(\omega) = 0$ for all $\omega \ne \tau'$;

End

Our algorithmic implementation framework has several important advantages. First, it is flexible to allow both system inputs and user-specified inputs. Second, it generalizes the SPR algorithm for 1-VNF systems to M-VNF, L-link systems. Even though the decomposition multipliers need to be computed in the decomposed model, once a decomposition node is selected, the original SPR algorithm is directly applied to generate the sample path. In the heuristic model, multiple SPR algorithms run in parallel to simultaneously generate sample paths independent of each other. All models embed the original SPR algorithm to solve the large-scale network problem under any general user-specified logical definition of overall system availability. This renders this framework highly scalable and adaptable to a wide range of availability criteria.

6. Computational Results

We conducted a comprehensive computational study of the proposed algorithmic framework. We assessed two key performance metrics in this study: statistical consistency in the downtime distributions estimated by the different algorithms and that of the benchmark integrated model, and the computational times involved in the estimation. We first present our experimental design, followed by a summary of key computational results. The online supplement provides more detailed experimental results that support the findings presented in this section. The comprehensive numerical experiments also yield ways to calibrate model parameters in the estimation process and perform a variety of sensitivity analyses with them.

6.1. Experimental Design

Since communication links can be treated as a special type of VNF (a link has only two states while a VNF can have $(n_m + k_m + 1)$ states), for simplicity and without loss of generality, we assume links are robust and only focus on VNF failures in this main experimental design. We further assume all VNF configurations are the same ($n_m = n$ and $k_m = k$ for $m = 1, \ldots, M$). We will relax this assumption and consider link failures in our algorithmic extension in Section 7.

We experimented with three values of the number of VNFs, $M = \{2, 5, 10\}$, and the number of primary VMs $n = \{100, 500, 1000\}$ in each VNF, representing small, medium and large sizes in the system configuration. We varied the number of backup VMs $k = \{3, 4, \ldots 19\}$ in different configurations of M and n to cover a wide range of service availability up to 5'9s. We chose the mean time to failure $MTTF = 3000$ and mean time to

repair $MTTR = 20$ based on our earlier extensive empirical studies (Du et al. 2015). So the VM failure and repair rates are $\lambda_m = 1/3000$ and $\mu_m = 1/20$. We normalized the service window $T = 1000$ and varied the value of Δt in our experiment, which affects the depth parameter ρ_{max}. In addition, the breadth parameter ε affects the number of sample paths required for the convergence of the estimated downtime distribution. We fixed $\varepsilon = 0.05$ in our experiments in order to focus on Δt and other key model parameters on computational performance.

We implemented our algorithms in R 3.3 (64-bit edition). An Intel(R) Core(TM) i7-4790 3.60 GHz processor equipped with 8GB of RAM was used for all experiments. We applied the Kolmogorov-Smirnov (KS) test to the downtime distributions obtained from different models for validation of statistical consistency.

6.2. Results

In the following discussion, starting from Exponential failure distribution, we first demonstrate the statistical consistency between the integrated and decomposed methods. Next, we show the superior performance of the decomposed model and the heuristic over the integrated approach under different failure distributional assumptions. Finally, we summarize our findings and discuss major insights.

6.2.1. Statistical Consistency. In this subsection, we demonstrate the statistical consistency among different model implementations assuming Exponential failure distribution. Figure 4 presents several examples to illustrate the statistical consistency among the downtime distributions produced by the integrated model, the decomposed model, and the heuristic based on the fixed Δt implementation. The integrated and decomposed variable Δt models also yield the same level of consistent distributions.

The Δt values shown in Figure 4 were the largest Δt under which the different methods yield consistent distributions, which we term as the maximum allowable Δt corresponding to a system configuration. For example, in the case of 5 VNFs, 1000 primary VMs, and 15 backup VMs, if we run different models using $\Delta t \leq 0.1$, the resulting distributions under different methods can simultaneously pass the pairwise KS tests, and thereby showing that they are statistically consistent with each other. The parameter value Δt=0.1 gained computational advantage than other smaller Δt's because the required number of steps to generate a random sample is smaller. However, if we further increase Δt beyond 0.1, then not all obtained distributions could pass the pairwise KS tests.

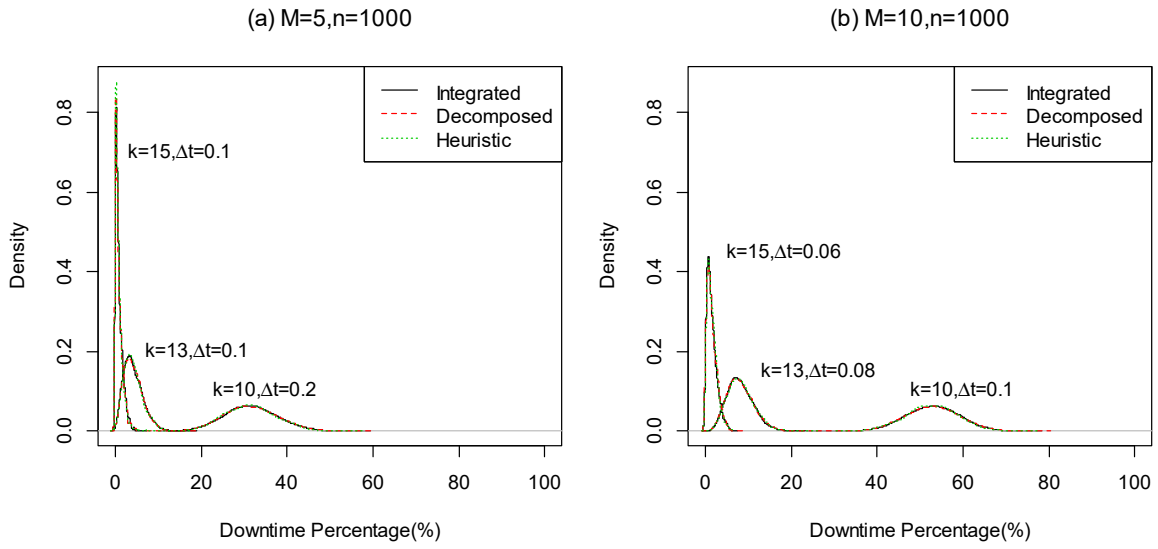

Figure 4 **Statistical Consistency of Downtime Distribution under the Integrated, Decomposed and Heuristic Models**

Under the same configuration of the number of VNFs (M) and VMs (n), as the number of backups (k) increases, both the mean and variance of the downtime distributions tend to decrease. Other things being equal, comparing plots (a) and (b) in Figure 4, we see that when the number of VNFs increases from 5 to 10, both the mean and variance of the downtime distribution increase. This is intuitive as the expected downtime would be more when the network is more complex. As a result, the maximum allowable Δt that guarantees statistical consistency decreases when the network becomes more complex (i.e., as the number of VNFs and the number of primary and backup VMs increase). Since both Δt and the required sample size for convergence would affect computational time, we next examine the algorithmic performance.

6.2.2. Computational Performance. In this subsection, we demonstrate the computational performance of different model implementations under different parametric settings. The online supplement provides the complete set of experiments and detailed computational results.

Figure 5 presents a performance comparison under representative system configurations: $(n = 100, k = 5)$, $(n = 500, k = 10)$, and $(n = 1000, k = 15)$ based on the Exponential failure distribution. Graphs for Weibull and Erlang distributions yield similar insights. They are presented in the online supplement. We compare five model implementations: the variable

Δt integrated model and decomposed model, the fixed Δt integrated model and decomposed model, and the fixed Δt heuristic. In the fixed Δt model implementation, the CPU time in the figure corresponds to the maximum Δt at which statistical consistency among all model implementations can be established based on KS tests.

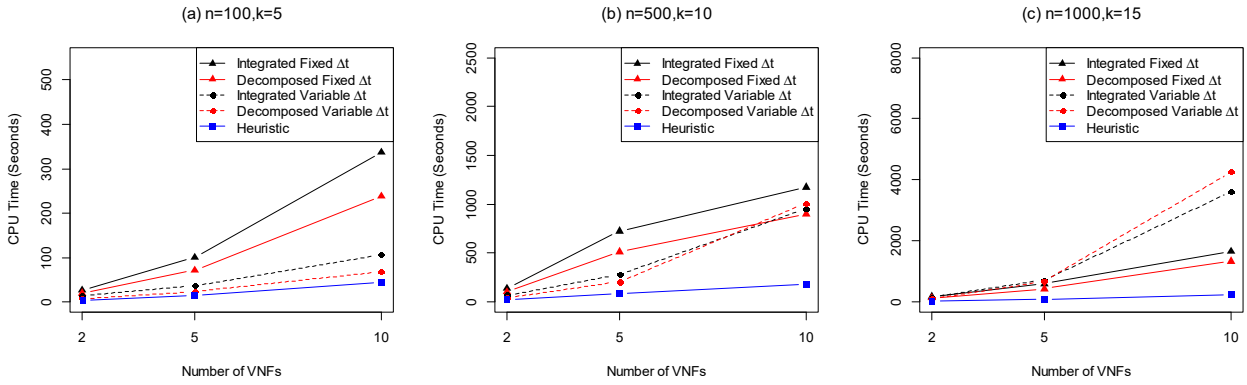

Figure 5 Illustration of Computational Performance under Exponential Distribution

We see that the CPU time (in seconds) increases as the number of VNFs increases. In terms of variable Δt model and fixed Δt model implementation, we find that the variable Δt model implementation performs better than the fixed Δt model when the number of primary VMs is small or when the number of VNFs is not large. As both the number of VNFs and VMs increase, fixed Δt model outperforms the variable Δt model. We believe the main reason is that the involved computation of Δt_{max} at each step of the state transition takes some CPU time. Overall, our results suggest that the variable Δt model outperforms the fixed Δt model in simple networks, but the fixed Δt model shows superior performance over variable Δt model in complex networks.

In terms of integrated model and decomposed model implementation, we find that the decomposed model consistently outperforms the integrated model. The variable Δt decomposed model is preferred in small and simple networks, while the fixed Δt decomposed model is preferred in large and complex networks. In general, the larger the number of VNFs, the larger the number of VMs, the higher the performance improvement. The same insights hold for Weibull and Erlang distributions as well (see Figure 3 in the online supplement).

In all cases, with careful calibration of Δt, fixed Δt heuristic approach achieves the best performance among all the model implementations. As shown in the figure, the time

saving is remarkable. As the number of VNFs increases, the required computational time under the heuristic implementation is only a fraction of the other model implementations. Overall, we see that both the decomposition model and the heuristic implementation are scalable to larger size systems.

6.2.3. Discussion. We have demonstrated that the proposed convex decomposition methodology can easily be applied to commonly used failure distributions, and can be used to quickly and accurately estimate system downtime. Regardless of the user choice of model implementation, the computational performance is affected by key system parameters as follows: First, under a specific configuration of (M, n, k), the computational time increases as Δt decreases. Second, as the number of backup VMs (k) increases, the mean downtime decreases, and both the required number of sample paths and computational time reduce. Finally, as the number of VNFs (M) and the number of primary VMs (n) in each VNF increases, both the required number of sample paths and computational time increase.

In terms of model selection, when the system has small number of VNFs and small number of primary VMs, we recommend the variable Δt model because of its optimal step size advantage when the feasible range of Δt is large. In contrast, when the system has large number of VNFs and large number of primary VMs, we recommend fixed Δt model because the feasible range of Δt's significantly reduces and the advantage of the variable Δt model diminishes. In the latter case, carefully calibrated heuristic outperforms the decomposition model, which in turn outperforms the integrated model. This shows scalability of our decomposition approach to achieve fast estimation of system downtime in larger and complex networks.

7. Adaptations and Extensions of the Algorithmic Framework

In the previous section, we have shown that our proposed decomposition algorithm is computationally efficient and can produce statistically consistent downtime distribution for a network of VNFs where: the VNFs have the same type of failure distribution, the network link connections are robust, and each VM is dedicated to a single VNF. In this section, we adapt and extend our algorithmic framework to various more complex scenarios that may arise in real world implementations. We consider three such extensions: (1) the VM failure distributions could vary across the VNFs, (2) network link connections may not be robust, and (3) VM sharing across different VNFs could occur in the implementation. Incorporating the specific requirements of each of these scenarios in the algorithmic framework, we

carried out extensive additional computational experiments. In all these experiments, we employed the fixed Δt model implementation. In the following, we discuss these algorithmic extensions, experimental designs and key conclusions. The online supplement provides more detailed experimental results.

7.1. Heterogeneous Failure Distributions

Incorporating different failure distributions for VNFs in the algorithmic framework is straightforward. This only requires initializing the algorithm with the appropriate failure distributions and their parameters for different VNFs at the input stage. The experimental design employed in this study is as follows. We experimented with 10 VNFs (M=10) characterized by different failure distributions: Exponential (4 VNFs), Weibull (3 VNFs), and Erlang (3 VNFs). We used the same shape parameters for the different failure distributions as in the main experiment described in Section 6. For the Weibull distribution, we considered both the decreasing failure rate (β=0.8) and increasing failure rate (β=1.2).

Similar to the main experimental design, we considered three levels of the number of primary VMs, n=$\{$100,500,1000$\}$. Under each n specification, we varied the number of backups k to cover a wide range of network availability up to 5'9s. In total, we experimented with 21 combinations of the (n, k) parameters. Under each of these configurations, we searched for the range of Δt values to ensure statistical consistency based on the Kolmogorov-Smirnov test across the integrated model, the decomposed model and the heuristic approach. We observe that the thresholds under different failure distributions remain in the same range as in the model with a single failure distribution. The required CPU time is also in the same range as in the main experiments (please refer to the online supplement for details). Figure 6 illustrates representative cases of the downtime distribution under different model implementations. Analogous behavior has been observed in all the experimental cases that were investigated. This shows that our algorithm is robust and it can achieve statistical consistency and computational efficiency under both homogeneous and heterogeneous failure distributions of VNFs.

7.2. Incorporation of Link Failures

In the main experiments described earlier, we have explicitly focused on VM failures without considering link failures. In this subsection, we extend our algorithm to address the impact of link failures on system downtime. Since VNFs may or may not need to communicate with each other, the topological structure of the network is defined by the specific

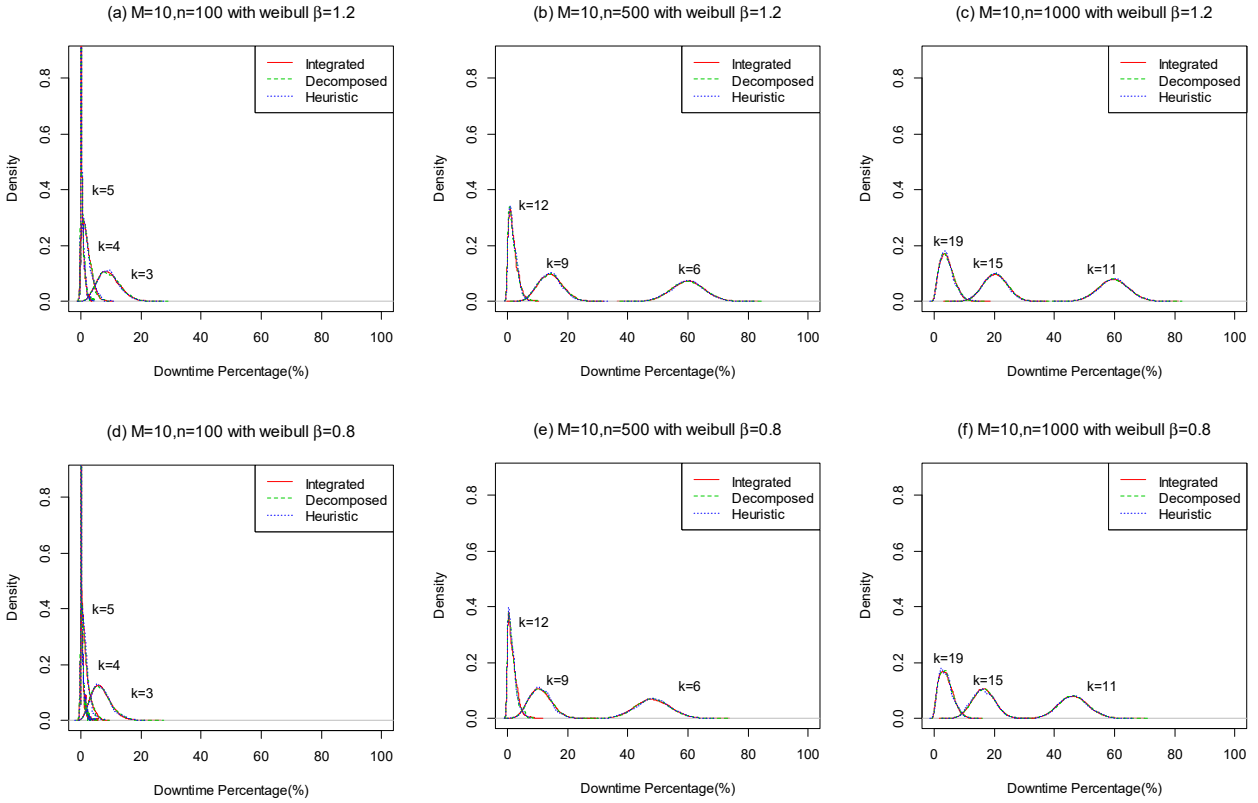

Figure 6 Illustration of Statistical Consistency of the Integrated, Decomposed and Heuristic Models

service requirement, which can be represented as a graph consisting of VNF nodes and the required communication links connecting different VNFs. Without loss of generality, we employed the following definition of availability in this study: the full network service is said to be available when (1) all the VNFs are concurrently available, AND (2) the required network connectivity is maintained. Accordingly, the system downtime at any time interval consists of either VNF-related downtime or connectivity-related downtime, or both. As mentioned earlier, this service definition can be generalized to any logical definition of availability as specified by a data center. To measure network connectivity, we first construct the *required reachability matrix* based on the original topological structure of the network service. Reachability is defined as the ability to reach from one VNF to another within a graph as per the connectivity requirement. At any time, if the *actual reachability matrix* and the required one are not the same, then it indicates a structural change of the original network connectivity, which will generate connectivity-related system downtime.

To maintain network connectivity, the service provider may consider: (1) to provide redundant links between any pair of VNFs that need to be connected (i.e., *link-level*

backup); (2) to utilize alternative, indirect network paths for connectivity when a direct link between two VNFs fail (i.e., *path-level* backup). There are two ways to extend our original algorithm to incorporate link-level backups. One way is to explicitly treat each link (including both the primary and backup links) as a two-state (on and off), special VNF (n=1,k=0). In this case, link connectivity is defined by the OR relationship among the primary and backup links connecting two communicating VNFs at the network level. The other way is to implicitly treat the link the same as a VNF. Since there is one primary link connecting two communicating VNFs, this link is equivalent to a (1,k) VNF design when k backup links are provided. Similar to the node-VNF, this link-VNF incurs downtime when more than k links fail simultaneously. These two approaches are equivalent and will yield the same estimated downtime distribution. The path-level backups are implemented in the algorithmic framework by using a comparison of the required reachability matrix and the actual matrix evaluated at any time. In this case, the availability of each link is modeled as in the link-level backup case.

For brevity and ease of presentation, we considered network configurations of M=$\{2,3,4,5\}$ VNFs, respectively. Using Exponential failures, we conducted three sets of experiments to systematically evaluate link failures in the presence of link-level and path-level backups, respectively. We first focus on fully connected networks to demonstrate the computational efficiency of the extended algorithm under various parameter values. We next study partially connected networks to investigate the differential effects of link failures and link-level/path-level backups on system downtime as the network density varies.

7.2.1. Fully Connected Networks. First, consider the 2-VNF network where no path-level backup is available. Similar to the main experiments, we evaluated the number of primary VMs at three levels: n=$\{100,500,1000\}$. For each n specification, we varied the number of backups k to cover a wide range of network availability up to 5'9s. In total, we experimented with 21 distinct combinations of (n,k) values. Since the 2-VNF network does not have any path-level backup, we considered the number of links L=$\{1,2,3\}$, representing no backup, one and two link-level backups, respectively. The link repair rate is kept at the same level as in our main experiments (i.e., μ_l=μ_E). The link failure rate parameter is tested at three levels: λ_l=$\{\lambda_E/5, \lambda_E, 5\lambda_E\}$, which represent fault-tolerant, moderately fault-tolerant, and fault-prone links.

As expected, the insights regarding statistical consistency and computational efficiency continue to hold with the consideration of link failure and backups. For illustration purpose, Figure 7 shows the downtime distributions for the (n=100, k=3) configuration with one-link, two-links and three-links in the 2-VNF network under the different failure rates, respectively. The downtime distribution in the main experiment without considering link failures is used as a baseline comparison.

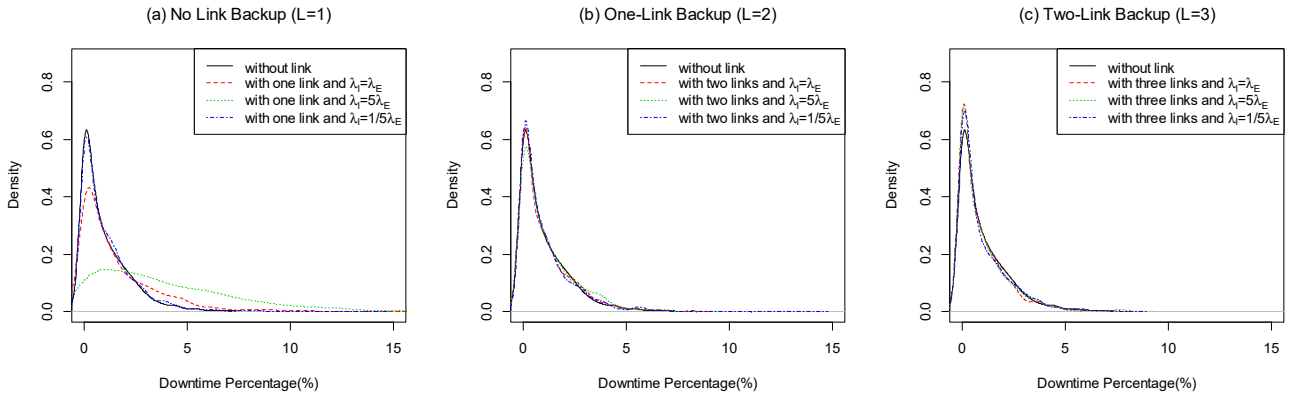

Figure 7 Impact of Link-Level Backup on Downtime Distribution under Different Failure Rates

Figure 7(a) shows that a relatively fault-tolerant link (λ_l=$\lambda_E/5$) behaves close to a fully robust link when no backup link is provided. However, as link failure rates increase ($\lambda_l = \lambda_E$ and $\lambda_l = 5\lambda_E$), the downtime increases, necessitating perhaps more than one backup link. Figures 7(b) and 7(c) indicate that the link connection becomes increasingly robust with more link-level backups, and perhaps a two-link backup would almost ensure full link robustness even with fault-prone connections.

Next, consider fully connected 3-, 4-, and 5-VNF networks (i.e., M=$\{3,4,5\}$). The corresponding number of links is L=$\{3,6,10\}$. We fixed the failure and repair rates at the same respective levels as in the main experiments. Since these networks naturally have more than one alternative path to connect a VNF to another VNF in the network, we did not consider link-level backups and just focused on path-level backups in this experiment to avoid the confounding effect. In the fully connected network, system downtime due to link failure happens only when there is at least one VNF that has no path connection with any of the other VNFs in the network. We observe that the maximum Δt that guarantees the statistical consistency across the different models is in the same range as in the main experiments and the required CPU time for convergence is slightly higher if we incorporate

link failures. Figure 8 illustrates the downtime distribution of a few typical examples when n=1000 and k={10,12,14} with Δt=0.1 where statistical consistency for all these experimental cases was obtained. Similar results have been obtained in all other experimental configurations.

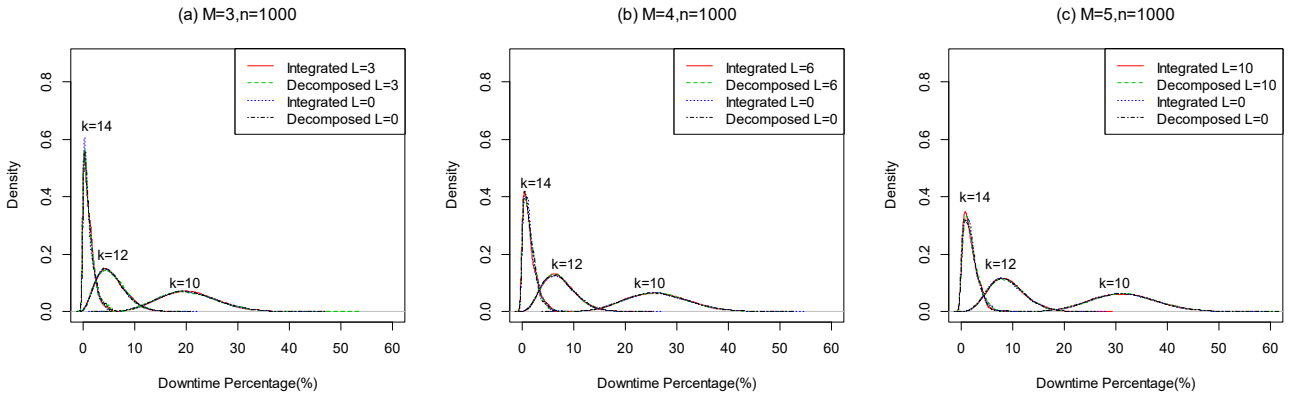

Figure 8 Impact of Path-Level Backup on Downtime Distribution in Multi-VNF Networks

Figure 8 shows that there is no significant difference in the downtime distribution between the multi-VNF network with path-level backup (i.e., L={3,6,10} for M={3,4,5} respectively) and the network without considering link failures (i.e., L=0 as in the main experiments). In all our experimental cases, even though individual links could be fault-prone, the natural path-level backup in multi-VNF networks is sufficient to guarantee the high level of path availability in most practical cases. The reason is that, the more nodes are in the network, more alternative paths exist to yield connectivity when a link fails. Therefore, the impact of link failures on the overall system availability is insignificant in fully connected networks.

7.2.2. Partially Connected Networks. In the previous subsection, we demonstrate that in fully connected networks, link failures hardly impact network availability. When networks are partially connected, link failures and backups may affect the network availability differently as the network connectivity levels vary. For example, link-level backups might be important in a less connected (i.e., low-density) network where not many natural path-level backups are available, and path-level backups might be sufficient in a more connected (i.e., high-density) network without the need to provide link-level backups. We next investigate their relationships in this experiment.

Formally, network connectivity is measured by the *network edge density*, which is defined as the ratio of the total number of edges in the network to the total number of possible edges in the corresponding fully connected network (Bollobás 1998). Therefore, as network edge density increases, the network becomes more connected. Expressing the density as a percentage, a fully connected network has edge density of 100%. In this experiment, we considered five levels of network edge density {20%, 40%, 60%, 80%, 100%}, two levels of link failure rate λ_l=$\{\lambda_E, 5\lambda_E\}$, and four levels of backups {0 link-level backup, 1 link-level backup, 2 link-level backup, and path-level backup}. In total, we had $5 \times 2 \times 4 = 40$ treatments.

Under each treatment condition, we randomly generated 30 network graphs. Since the size of the network does not affect the major qualitative insights in our pilot runs of the experiment, we fixed M=5, n=100 and k=5 for illustration purposes. We also fixed Δt=0.1. We ran the algorithm to get the overall system downtime distribution of each network in the experiment. Table 1 reports the mean and standard deviation of the mean downtime from the estimated downtime distribution based on the 30 randomly generated networks at different density levels.

Table 1 Mean and Standard Deviation (in brackets) of the Mean Downtime under Different Network and Backup Configurations

Network Density	$\lambda_l = \lambda_E$ **(Moderately Fault-Tolerant Links)**				$\lambda_l = 5\lambda_E$ **(Fault-Prone Links)**			
	Link-Level Backups			Path-Level Backups	Link-Level Backups			Path-Level Backups
	0	1	2		0	1	2	
20%	0.0140 (0.0049)	0.0004 (0.0001)	0.0004 (0.0001)	0.0136 (0.0045)	0.0654 (0.0224)	0.0024 (0.0001)	0.0004 (0.0001)	0.0650 (0.0221)
40%	0.0272 (0.0055)	0.0006 (0.0001)	0.0004 (0.0001)	0.0157 (0.0091)	0.1263 (0.0223)	0.0047 (0.0007)	0.0005 (0.0000)	0.0755 (0.0410)
60%	0.0390 (0.0043)	0.0006 (0.0001)	0.0004 (0.0001)	0.0049 (0.0043)	0.1783 (0.0197)	0.0066 (0.0001)	0.0006 (0.0001)	0.0245 (0.0211)
80%	0.0525 (0.0046)	0.0007 (0.0001)	0.0003 (0.0001)	0.0011 (0.0022)	0.2316 (0.0194)	0.0085 (0.0010)	0.0007 (0.0001)	0.0042 (0.0103)
100%	0.0629 (0.0010)	0.0008 (0.0001)	0.0004 (0.0001)	0.0004 (0.0001)	0.2749 (0.0005)	0.0104 (0.0003)	0.0007 (0.0001)	0.0004 (0.0001)

Table 1 yields several interesting observations. First, consider the link-level backup strategy. The mean downtime is the highest when no backups are provided and significantly decreases when link-level backups are provided. This can be observed in both the case of fault-tolerant and fault-prone links, and this effect of additional backups is more marked

when the links are fault-prone. Similarly, the mean downtime increases with network density in all link-level backup cases studied (0, 1 and 2 backups) for both fault-tolerant and fault-prone links. This shows that as the network connectivity increases, more links are prone to failure, thus the effects of link failure on increasing downtime become more significant, and consequently, the remedial effects of link-level backups also become more significant.

Second, consider the path-level backup strategy. We observe that the mean downtime is relatively higher in low-density than high-density networks. In particular, there is a slight increase in the mean downtime when density is increased from 20% to 40%, and then the downtime decreases significantly with further increase in the density. This can be interpreted as follows. As the density increases from 20% to 40%, more links are subject to failure and there are not enough alternative paths to compensate this by yielding the required connectivity; as the density increases beyond 40%, more alternate paths come into play, overriding the increased effects of link failures on overall system downtime. This effect of path-level backup on mean downtime with increasing network density is also corroborated by similar trends in CPU times involved. This is discussed in section 7.4.

Third, comparing the effects of link-level and path-level backups we see that, when the network density is low (20%), the effect of path-level backup is as weak as no-backup. When the network density is very high (100%), path-level backup is as good as providing two or more link-level backups. Therefore, we conclude that link-level backup dominates path-level backup when the network density is low, whereas path-level backup could dominate link-level backup when the network density is high.

Increasing downtime not only leads to potential violation of SLA but also causes larger sample sizes for the sample path randomization algorithm to converge; consequently, it leads to higher CPU time to obtain statistically consistent distribution of the downtime. The details of this observation are given in Table 11 and its following discussion in the online supplement. Providing backups are thus very important under these situations. To achieve high network availability (e.g., 99.9% or above), one link-level backup is sufficient for fault-tolerant links, and two or more link-level backups are needed when links are fault-prone. Path-level backups can only achieve high network availability when the network is almost fully connected.

Our experiments demonstrated the tradeoffs among availability, performance and cost of backup provisioning in real data center operations. In terms of availability, link-level backups and path-level backups are somewhat substitutable in practice. In terms of performance, link-level backup may have higher quality and speed in data transfer than path-level backups due to direct communication. However, link-level backups incur additional resource provisioning cost while path-level backups do not. Therefore, the service provider should tradeoff these three factors in its backup provisioning strategy, especially recognizing cost savings versus potential performance degradation under path-level backup. In general, the service provider will be able to take advantage of the naturally existing path-level backups available in a network to guarantee high service availability under highly connected network configurations. So, in general, there is no need to provide additional link-level backups unless the following unlikely conditions occur: (1) the links are extremely unreliable and the network is relatively less connected, or (2) link capacities in alternative paths may be too limited to provide the same level of data transfer rates as available in the direct links between VNFs. In this latter case, performance could be degraded although downtime may not be incurred. Modeling these behaviors as part of downtime estimation is an important area of research that is beyond our current scope. Large-scale implementation and comprehensive investigation of the relationship between node and link failures is also beyond the current scope of this research. We recommend these for future investigations.

In summary, network availability is affected by three broad layers of failures. The first layer is the VM-level failures, the second layer is VNF-level failures, and the third layer is the network-level failures which consist of either VNF-related or connectivity-related failures. Link failures can be easily factored into the network availability logic. The overall reachability among network components can be defined according to any network topology, and the availability logic can be modeled based on their AND/OR relationships that define the whole network availability. In this study, we have demonstrated link failures between VNFs in a network of VNFs (inter-VNF links). Furthermore, it is possible to envision connectivity among the VMs hosting a VNF and link failures among these VMs leading to VNF-level downtime. The logic of the proposed framework can be extended to these intra-VNF downtime characterizations and is recommended for future studies. Finally, hierarchies of VNFs in network design comprising of VNFs such as application nodes, communication nodes and power nodes are possible. Specializing the proposed downtime

estimation framework to these architectures are important and viable avenues for future research.

7.3. Adaptation to VM Sharing across VNFs

In this extension to the framework, we relax our original assumption that each VM independently supports only one VNF. Accordingly, we consider the situation where a VM can either be *dedicated* to a VNF or be *shared* by multiple VNFs. Although the impact of shared VMs on network service availability is a complex issue that highly depends on the network structure and design, the proposed framework can be extended to any design configuration of VM sharing among the VNFs. For simplicity of presentation and without loss of generality, we illustrate this framework extension by assuming that any shared VM is fully shared by all the VNFs. Further extensions to other more complex sharing structures where a shared VM is shared by only a subset of VNFs (i.e., partial sharing) can easily be obtained from this strategy, and we discuss this subsequently.

Consider a VNF in an M-VNF network. Denote its number of shared VMs as n_s and the number of dedicated VMs as n_d. So the number of primary VMs that support the VNF is $n = n_s + n_d$. Without loss of generality, we assume all backup VMs are dedicated VMs, and each VNF is supported by n primary VMs and k backups. Recall that all the VMs supporting a VNF are dedicated in the original framework described earlier. Therefore, in the original framework, the state space of the Markov chain in the integrated model is given by $i_I = 0, 1, \ldots, M(n+k)$, and we maintain a separate Markov chain $i_m = 0, 1, \ldots, (n+k)$ for each VNF m in the decomposed model. In the extended framework, since a proportion of VMs are fully shared by all VNFs while the rest are dedicated to specific VNFs, the state space of the integrated Markov chain is smaller than that of the original model: $i_I = 0, 1, \ldots, n_s + M(n_d + k)$. In the decomposed model, we still main a separate Markov chain for each VNF with its dedicated VMs. The state space of the dedicated chain is therefore $i_m^d = 0, 1, \ldots, (n_d + k)$, where the superscript on i_m^d is used to indicate that the VMs are all dedicated. Compared with the original decomposed chain, this chain becomes shorter because $n_d < n$. In addition, we add a new Markov chain for the shared VMs. The state space of the shared chain is $i_s = 0, 1, \ldots, n_s$. The current state for VNF m is thus determined by the shared and dedicated VM nodes as $i_m = i_s + i_m^d$. Similar to the original framework, i_m determines if VNF m is up or down at any given time, and the overall system is considered to be down if at least one VNF is down at that time.

The implementation of this extension to incorporate VM sharing in the decomposition algorithm is straightforward. We simply treat the Markov chain associated with the shared VMs as an additional VNF chain. Hence, there are $(M+1)$ decomposed Markov chains in this extension. The decomposition multipliers can be computed in exactly the same way as in the original model.

The design of the computational experiments with this extension is as follows. We performed experiments based on the 2-, 5-, and 10-VNF structures (i.e., M={2,5,10}). Similar to the original experiments and without loss of generality, we assumed Exponential failures for the VMs. The number of primary VMs was evaluated at three levels: n={100,500,1000}, For each n specification, we varied the number of backups k to cover a wide range of network availability up to 5'9s, resulting in 21 combinations of the (n,k) parameters. In addition, we varied the proportion of VM sharing in a VNF at three levels of the number of primary VMs, $n_s = \{20\%n, 50\%n, 80\%n\}$, representing low, medium, and high levels of sharing. In total, we evaluated $(3 \times 21 \times 3) = 189$ experimental configurations. All experiments show that the extended algorithm produces statistically consistent downtime distributions between the integrated and decomposed models, and is computationally efficient. Figure 9 presents three examples using M=10 to illustrate the effects of changing level of VM sharing on the system downtime distributions. Similar behaviors have been observed across all experiments.

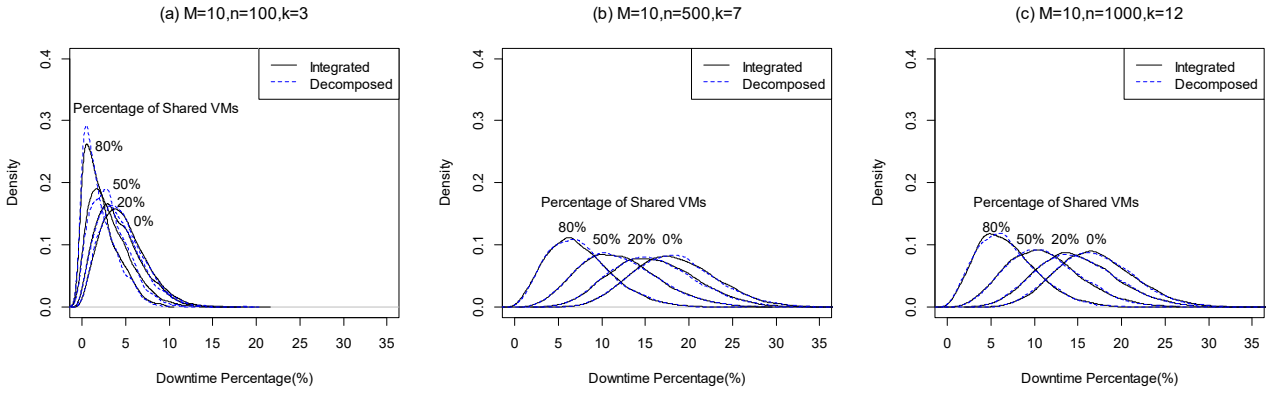

Figure 9 Effects of VM Sharing on Downtime Distribution

We see that both the mean downtime and the variance decrease as the percentage of VM sharing increases. Intuitively, this is because we define the network availability as the concurrent availability of all the VNFs. Accordingly, since the overall network is down if

any VNF is down, and a VM failure of a critical VNF (i.e., a VNF that has exactly n VMs working just before the failure event) would cause the overall network to be down, a shared VM failure has the same effect as a dedicated VM failure on the availability of the critical VNF. However, a shared VM repair contributes to availability of all VNFs, which advances the number of available VMs by 1 to all the VNFs. As a result, the repair of a shared VM is more beneficial to the overall network availability than a dedicated VM repair. Therefore, the collective benefit of a shared VM repair is higher than the cost of shared VM failure. Consequently, the network availability increases when the percentage of VM sharing increases.

As noted earlier, the case of fully shared VMs is easily implemented by simply adding one additional Markov chain for the shared VMs in which the changes of state would affect all VNFs. However, in the case of partial sharing, we need to treat each shared VM as a separate Markov chain in the decomposition process; in this situation, the changes of state would affect only the associated VNFs that share that VM. Our algorithm is easily scalable by adding more Markov chains based on the sharing structures.

In summary, we have shown in these extensions that: (1) our algorithm can easily be adapted to different failure distributions, (2) can easily incorporate link failures and network structures, and (3) can consider different levels of VM sharing across VNFs. Importantly, we demonstrate that statistical consistency can be obtained in all these cases and the computational load for both the integrated model and decomposed model under the extended framework are in similar range as in the main experiments. These additional experiments further demonstrate that the proposed algorithmic framework is not only scalable, adaptable and computationally efficient, but also easily extensible to the several practical system requirements and modifications studied.

7.4. Computational Performance

In this subsection, we demonstrate the computational performance of our extended algorithms. Figure 10 comprising of plots (a)-(f) presents a performance comparison of the integrated model and the decomposed model under the three model extensions studied. The online supplement provides the detailed computational results.

Under heterogeneous failure distributions, plot (a) shows that the CPU time (in seconds) increases as the number of primary and backup VMs increases. The required computational times incorporating the increasing Weibull failure rate (i.e., β=1.2) are relatively higher

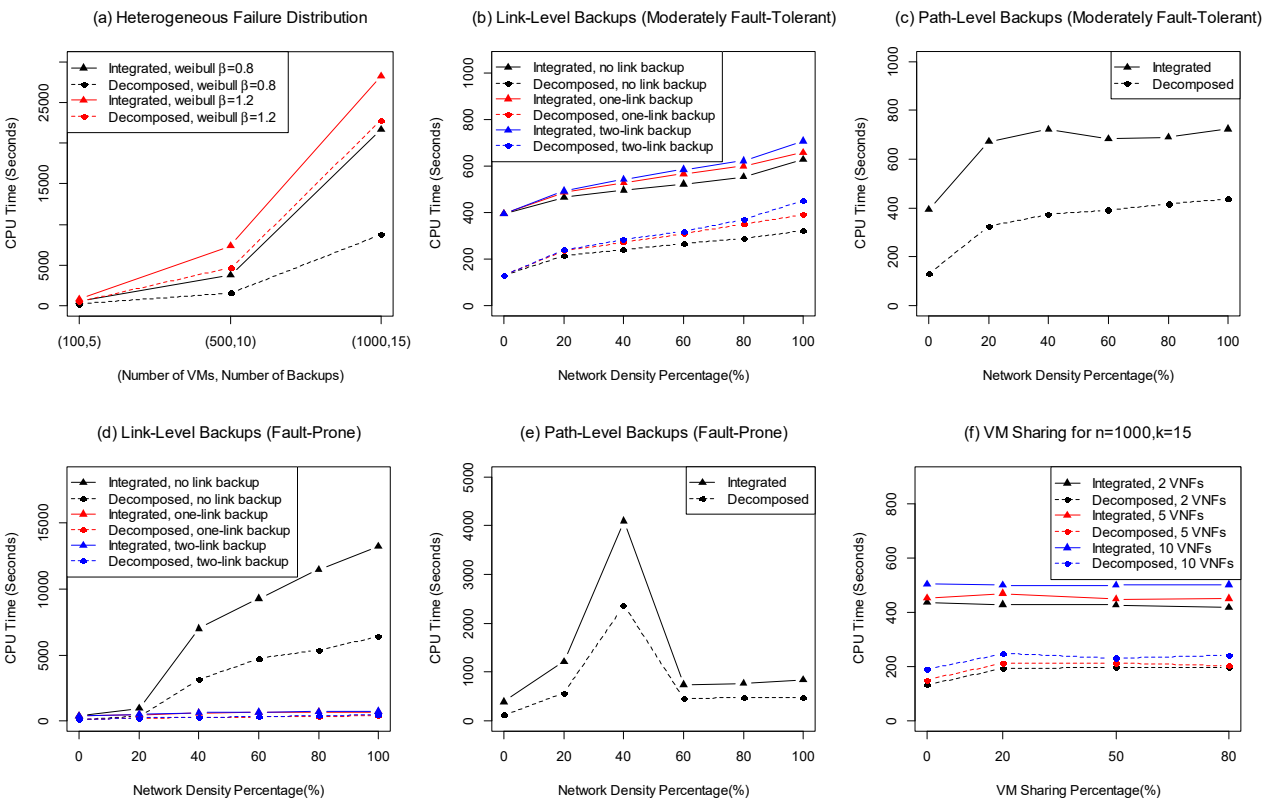

Figure 10 **Illustration of Computational Performance for the Extended Algorithms**

than the decreasing Weibull failure rate (i.e., β=0.8). Overall, the decomposed model has achieved an average of 45.4% time saving than the integrated model in all our experimental configurations under heterogeneous failure distributions. The CPU time savings of the decomposed model over the integrated model increases with the complexity of the system configuration.

Plots (b)-(e) illustrate the impacts of link failures on computational performance of the proposed algorithm. We considered the impact on three dimensions: link reliability (moderately fault-tolerant, fault-prone), network density (0%-100%), and type of backups (link, path). For moderately fault-tolerant links, plots (b) and (c) show that the CPU time generally increases with the number of backup links and the network density. In contrast, the effects are more significant and non-monotonic for fault-prone links. For link-level backup, we observe significantly higher CPU time when there is no link-level backups than otherwise (see plot (d)). For path-level backup, we observe that the CPU time first increases significantly as the network density increases, but then decreases significantly after a level (see plot (e)). The main reason for the initial increase in computational load would be due to the increase in the number of unreliable links that would incur significant

system downtime, leading to greater CPU time for estimation; similarly, the decrease in CPU time beyond reaching a level of network density would be due to the stabilizing effect of the additional path-level backups available on the overall system downtime. Overall, the decomposed model has achieved an average of 43.5% time saving than the integrated model in the extended algorithm incorporating link failures.

Finally, plot (f) shows the effect of VM sharing on computational performance. The CPU time increases in the number of VNFs but does not seem to be significantly affected by the percentage of VM sharing. Overall, the decomposed model has achieved an average of 55.5% time saving than the integrated model under all our experimental configurations when considering VM sharing. The performance gain of the decomposed model over the integrated model increases with the complexity of the system configurations.

In summary, our proposed decomposition strategy is robust, both in terms of algorithm design and computational performance, to handle various types of model extensions. All our experimental results strongly demonstrate a consistently superior performance of the decomposition strategy over the integrated modeling approach.

8. Conclusion

In this study, we define and measure network service availability in virtual infrastructures consisting of a number of VNFs and communication links. We have proposed a convex decomposition framework to estimate the transient downtime under different system configurations and failure distributions. We first theoretically show the statistical consistency between the proposed convex decomposition method and the integrated model. We then empirically demonstrate the computational efficiency of the decomposition approach over the integrated method. We find the computational time increases in the number of VNFs and the number of primary VMs, and decreases in the number of backup VMs. As the system becomes more complex, the heuristic and decomposition models show superior performance over the integrated approach, especially under the fixed Δt implementation.

In addition to computational efficiency, the proposed decomposition strategy has several other important advantages. First, due to the decomposable structure and low dimensionality, our algorithm is highly scalable to solve large network problems. Second, because each system component independently generates its own sample path, the sample paths generated by the decomposed approach are reusable for other logical definitions of network

availability. The modular structure enables the existing network model to be reused as building blocks for more complex services. Finally, the proposed decomposition framework is adaptable and generalizable to accommodate various practical system requirements such as heterogeneous failure distributions, VM sharing across VNFs, networks with various degrees of connectivity, and hierarchical network structures. Although we only demonstrate our proposed methods at the virtual server nodes level, we can easily add other types of nodes such as power nodes and switch nodes because the network dependence and independence can be modeled as AND/OR logical relationships, respectively. Logical definition of network availability can be adapted based on different network topologies.

Because of the above advantages of the proposed framework, especially the scalability and computational efficiency of the decomposed approach, we provide practical solutions to the cloud service providers to quickly estimate downtime distribution, enabling them to optimize resource utilization on the fly. Cloud resource management requires policies and decisions that minimize the cost of providing services at guaranteed level of system performance. Due to the low computational cost, our algorithms can be run frequently to estimate the network downtime distribution in real time. The service provider may dynamically adjust the allocation of backup resources to minimize the estimated expected backup resource provisioning cost and the contract violation penalty cost. Given the cloud service provider's dynamic resource management capability, optimizing SLA contract design would be an interesting future research direction.

Acknowledgments

We sincerely thank the Area Editor, the Associate Editor, and the reviewers for their many helpful suggestions which greatly contributed to this research. Jin Li is the corresponding author and would like to thank the support from the National Natural Science Foundation of China [No. 71771184].

Appendix. Notation Table

Notation	Definition
M	Total number of VNFs
L	Total number of communication links
n_m	The number of primary VMs required by the mth VNF, $m = 1, \ldots, M$
k_m	The number of backup VMs for the mth VNF, $m = 1, \ldots, M$
$i_m = \{0, \ldots, n_m + k_m\}$	Current state of VNF m (the number of failed VMs in the mth VNF), $m = 1, \ldots, M$
$c_l = \{on, off\}$	Current state of communication link l (either on or off) , $l = 1, \ldots, L$

Notation	Definition
$\bar{c}_l$	Opposite state of c_l, $l=1,\ldots,L$
$\boldsymbol{IC}=[i_1,\ldots,i_M,c_1,\ldots,c_L]'$	The current $(M+L)$-dimensional vector of the system states
$\mu_m=\frac{1}{MTTR_m}$	Component m's repair rate under Exponential, Weibull and Erlang distribution, $m=1,\ldots,M+L$
$\lambda_m=\frac{1}{MTTF_m}$	Component m's failure rate under Exponential distribution, $m=1,\ldots,M+L$
$\lambda_m(t)$	Component m's failure rate at time t under Weibull and Erlang distribution, $m=1,\ldots,M+L$
α_m, β	Scale and shape parameters in Weibull distribution, $m=1,\ldots,M+L$
e_m, γ	Scale and shape parameters in Erlang distribution, $m=1,\ldots,M+L$
r_{c_l}	Link l's failure (or repair) rate when the state is $c_l=\{on,off\}$, $l=1,\ldots,L$
δ_m,δ_l	The VNF and link's convex decomposition multiplier, $m=1,\ldots,M$ and $l=1,\ldots,L$
$P_{\boldsymbol{IC}}$	Probability of remaining in the same state in the integrated Markov chain model
$P^m_{i_m,i_m-1}$	State transition probability that VNF m repairs one server in the integrated Markov chain, $m=1,\ldots,M$
$P^m_{i_m,i_m+1}$	State transition probability that VNF m fails one server in the integrated Markov chain, $m=1,\ldots,M$
$P^l_{c_l,\bar{c}_l}$	Probability that link l transits out of state c_l in the integrated Markov chain, $l=1,\ldots,L$
$\tilde{P}^m_{i_m,i_m+1}$	State transition probability that VNF m fails one server in the decomposed Markov chain, $m=1,\ldots,M$
$\tilde{P}^m_{i_m,i_m-1}$	State transition probability that VNF m repairs one server in the decomposed Markov chain, $m=1,\ldots,M$
$\tilde{P}^m_{i_m,i_m}$	Probability that VNF m remains in the same state in the decomposed Markov chain, $m=1,\ldots,M$
$\tilde{P}^l_{c_l,\bar{c}_l}$	Probability that link l transits out of state c_l in the decomposed Markov chain, $l=1,\ldots,L$
T	Service window
$t\in[0,T]$	Real time of the system within the service window
$\Delta t=\{\Delta t_{fix},\Delta t_{max}\}$	Step size in the Markov chain state transition when generating a random sample path
$\rho=\{1,\ldots,\rho_{max}\}$	Index of time intervals in the service window T, where ρ_{max} is the total number of steps to complete a sample path (the depth parameter)
ε	Convergence threshold for sample path generation (the breadth parameter)
S	Total number of sample paths (sample size)
d_s	A random sample path corresponding to the sth generation, $s=1,\ldots,S$
$A_{m\rho}$	Binary variable indicating VNF availability in the ρth time interval
$B_{l\rho}$	Binary variable indicating link availability in the ρth time interval
$I_\rho=\mathcal{L}(A_{m\rho},B_{l\rho})$	Binary variable indicating network availability based on system availability logic $\mathcal{L}$ in the ρth time interval.
$\tau_s=1-\sum_{(t<T)}I_\rho\Delta t/T$	Percentage of time the network is down for sample path s, $s=1,\ldots,S$

References

Bollobás B (1998) *Modern Graph Theory* (Springer).

Buyya R, Garg SK, Calheiros RN (2011) SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. *IEEE International Conference on Cloud and Service Computing (CSC)*, 1–10.

Chowdhury NMK, Boutaba R (2010) A survey of network virtualization. *Computer Networks* 54(5):862–876.

Du AY, Das S, Yang Z, Qiao C, Ramesh R (2015) Predicting transient downtime in virtual server systems: An efficient sample path randomization approach. *IEEE Transactions on Computers* 64(12):3541–3554.

ETSI (2013a) Network functions virtualisation NFV: Architectural framework. Technical report, ETSI Industry Specification Group.

ETSI (2013b) Network functions virtualisation NFV: Use cases. Technical report, ETSI Industry Specification Group.

ETSI (2014) Network function virtualization NFV: Management and orchestration. Technical report, ETSI Industry Specification Group.

Fu S (2010) Failure-aware resource management for high-availability computing clusters with distributed virtual machines. *Journal of Parallel and Distributed Computing* 70(4):384–393.

Gill P, Jain N, Nagappan N (2011) Understanding network failures in data centers: measurement, analysis, and implications 41(4):350–361.

Han B, Gopalakrishnan V, Ji L, Lee S (2015) Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine* 53(2):90–97.

IHS (2015) NFV hardware, software and services report. Technical report, Infonetics.

Kaczynski WH, Leemis LM, Drew JH (2012) Transient queueing analysis. *INFORMS Journal on Computing* 24(1):10–28.

Lu P, Ravindran B, Kim C (2012) VPC: Scalable, low downtime checkpointing for virtual clusters. *IEEE 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 203–210.

Machida F, Kawato M, Maeno Y (2010) Redundant virtual machine placement for fault-tolerant consolidated server clusters. *IEEE Network Operations and Management Symposium (NOMS)*, 32–39.

Martini B, Paganelli F (2016) A service-oriented approach for dynamic chaining of virtual network functions over multi-provider software-defined networks. *Future Internet* 8(2):24.

Mijumbi R, Serrat J, Gorricho JL, Bouten N, De Turck F, Boutaba R (2016) Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials* 18(1):236–262.

Mu S, Su M, Gao P, Wu Y, Li K, Zomaya AY (2015) Cloud storage over multiple data centers. *Handbook on Data Centers*, 691–725 (Springer).

Parpas P, Ustun B, Webster M, Tran Q (2015) Importance sampling in stochastic programming: A Markov chain Monte Carlo approach. *INFORMS Journal on Computing* 27(2):358–377.

Passacantando M, Ardagna D, Savi A (2016) Service provisioning problem in cloud and multi-cloud systems. *INFORMS Journal on Computing* 28(2):265–277.

Raad P, Secci S, Phung DC, Cianfrani A, Gallard P, Pujolle G (2014) Achieving sub-second downtimes in large-scale virtual machine migrations with LISP. *IEEE Transactions on Network and Service Management* 11(2):133–143.

Sharkh MA, Jammal M, Shami A, Ouda A (2013) Resource allocation in a network-based cloud computing environment: design challenges. *IEEE Communications Magazine* 51(11):46–52.

Siewiorek DP, Swarz RS (2017) *Reliable computer systems: design and evaluation* (Digital Press).

Silva E, Gail H (1986) Calculating cumulative operational time distributions of repairable computer systems. *IEEE Transactions on Computers* 35(4):322–332.

Silva E, Gail H (1989) Calculating availability and performability measures of repairable computer systems using randomization. *Journal of the ACM* 36(1):171–193.

Sun C, Bi J, Zheng Z, Hu H (2016) SLA-NFV: an SLA-aware high performance framework for network function virtualization. *Proceedings of the 2016 conference on ACM SIGCOMM*, 581–582 (ACM).

Xiao A, Wang Y, Meng L, Qiu X, Li W (2013) Topology-aware remapping to survive virtual networks against substrate node failures. *IEEE 15th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 1–6.

Yang Y, Zhang Y, Wang AH, Yu M, Zang W, Liu P, Jajodia S (2013) Quantitative survivability evaluation of three virtual machine-based server architectures. *Journal of Network and Computer Applications* 36(2):781–790.

Yu H, Anand V, Qiao C (2012) Virtual infrastructure design for surviving physical link failures. *The Computer Journal* 55(8):965–978.

Yu H, Anand V, Qiao C, Sun G (2011) Cost efficient design of survivable virtual infrastructure to recover from facility node failures. *IEEE International Conference on Communications (ICC)*, 1–6.

Yu H, Qiao C, Anand V, Liu X, Di H, Sun G (2010) Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures. *IEEE Global Telecommunications Conference (GLOBECOM)*, 1–6.

Yuan S, Das S, Ramesh R, Qiao C (2014) Availability-aware resource provisioning, pricing, and allocation adjustment in the cloud. *Conference on Information Systems and Technology, San Francisco, CA*.