Doctoral Thesis

Reliable Resource Allocation Models in Network Virtualization

Fujun HE

Graduate School of Informatics, Kyoto University

September 2020

Preface

Network virtualization has been introduced as a key role in the next-generation networking paradigm to fend off the ossification of traditional networks. By leveraging the technologies of computer virtualization, network function virtualization, and software-defined networking, a platform with network virtualization provides virtualized resources of computing, functionality, and networking to users in a dynamic manner. While network virtualization leads to a more flexible and efficient network, it brings challenges for network management, one of which is how to efficiently allocate resources with satisfying different requirements. In addition, as the adaptation of network virtualization in different application archetypes is an increasing trend, the reliability of an environment with network virtualization has become a major concern. Resource allocation with protection strategies dealing with the reliability issue is an essential requirement for network virtualization. This thesis studies five specific problems about reliable resource allocation in network virtualization, each of which focuses on a typical application scenario, to achieve a flexible, cost-effective, and dependable network virtualization environment.

Firstly, this thesis proposes a primary and backup resource allocation model that provides a probabilistic protection guarantee for virtual machines against multiple failures of physical machines in a cloud provider to minimize the required total capacity. The probability that the protection provided by a physical machine does not succeed is guaranteed within a given number. Providing the probabilistic protection can reduce the required backup capacity by allowing backup resource sharing, but it leads to a nonlinear programing problem in a general-capacity case against multiple failures. This work applies robust optimization with extensive mathematical operations to formulate the resource allocation problem as a mixed integer linear programming problem, where

capacity fragmentation is suppressed. This work proves the NP-hardness of considered problem. A heuristic is introduced to solve the optimization problem. The results reveal that the proposed model saves about one-third of the total capacity in the examined cases; it outperforms the conventional models in terms of both blocking probability and resource utilization.

Secondly, this thesis proposes a backup computing and transmission resource allocation model for virtual networks with the probabilistic protection against multiple facility node failures. Backup transmission resource allocation is incorporated, where the required backup transmission capacity can affect the required backup computing capacity. This work analyzes backup transmission resource sharing in the case of multiple facility node failures to compute the minimum required backup transmission capacity. A heuristic algorithm is introduced to solve the problem; especially, several techniques based on graph theory are developed to handle the problem with full backup transmission resource sharing. The results observe that the proposed model outperforms a baseline with dedicated protection for computing resource.

Thirdly, this thesis proposes a backup resource allocation model for middleboxes with considering both failure probabilities of network functions and backup servers. This work takes the importance of functions into account by defining a weighted unavailability for each function. This work aims to find an assignment of backup servers to functions where the worst weighted unavailability is minimized. This work formulates the proposed model as a mixed integer linear programming problem. This work proves that the considered problem is NP-complete. This work develops three heuristic algorithms with polynomial time complexity to solve the problem. This work analyzes the approximation performances of different heuristic algorithms with providing several lower and upper bounds. This work presents the competitive evaluation in terms of deviation and computation time among the results obtained by running the heuristic algorithms and by solving the mixed integer linear programming problem. The results show the pros and cons of different approaches. With the analyses, a network operator can choose an appropriate approach according to the requirements in specific applications.

Fourthly, this thesis proposes an unavailability-aware backup allocation model with the shared protection to minimize the maximum unavailability

among functions. The shared protection allows multiple functions to share the backup resources, which leads to a complicated recovery mechanism and makes unavailability estimation difficult. This work develops an analytical approach based on the queueing theory to compute the middlebox unavailability for a given backup allocation. The heterogeneous failure, repair, recovery, and waiting procedures of functions and backup servers, which lead to several different states for each function and for the whole system, are considered in the queueing approach. This work introduces a simulated annealing heuristic to solve the problem based on the developed analytical approach. The results reveal that, compared to a baseline model, the proposed model reduces the maximum unavailability 16% in average in the examined scenarios.

Fifthly, this thesis proposes a master and slave controller assignment model against multiple controller failures in software-defined networks with considering propagation latency between switches and controllers. Given assigned controllers for a switch, the master controller in each failure case is automatically specified based on a low latency first policy. This work defines three objectives to be optimized, which lead to three different problems. This work proves that the adopted policy achieves the optimal objectives for considered problems. This work formulates the proposed model with different goals as three mixed integer linear programming problems. This work proves the NP-completeness for all the three problems. A greedy algorithm with polynomial time complexity is developed; this work shows that it provides a 1/2-approximation for the case without the survivability guarantee constraint. The numerical results observe that the proposed model obtains the optimal objective value with the computation time about 10^2 times shorter than that of a baseline that introduces decision variables to determine the master controllers.

This thesis is organized as follows. Chapter 1 introduces the background of reliable resource allocation in network virtualization. Chapter 2 investigates the related works in literature. Chapter 3 presents the robust optimization model for virtual machine allocation. Chapter 4 introduces the probabilistic protection model for virtual networks. Chapter 5 develops the backup resource allocation model for network functions. Chapter 6 presents the unavailability-aware shared backup allocation model. Chapter 7 describes the controller assignment model for switches. Finally, Chapter 8 concludes this thesis.

Acknowledgements

This thesis is the summary of my doctoral study at Kyoto University, Kyoto, Japan. I am grateful to a large number of people who have helped me to accomplish this work.

First of all, I would like to express my sincere gratitude to my advisor, Professor Eiji Oki, for his mentorship, guidance, and encouragements. Especially, without the inspiration from Professor Oki, I could not find that doing research can be such interesting. By working under the supervision of Professor Oki, I received an invaluable experience that helped me to shape my academic and professional skills.

Second, I would like to thank Professor Ryoichi Shinkuma for the insightful discussions. I would like to express my appreciation to Professor Takehiro Sato for his advisements and helps.

I would like to thank Professor Masahiro Morikura and Professor Hiroshi Harada for being part of my judging committee and providing their precious comments to improve my thesis.

I would also like to thank Ms. Mariko Tatsumi and all members in Oki lab for their kind helps in my research and life.

Finally, I want to deeply thank my parents, grandparents, and beloved girl, Shihan. Without their constant support and encouragements, I would never achieve anything.

Contents

List of Figures

List of Tables

Notations

Notation	Description
r_{ij}^{e}	Given parameter indicating requested capacity of existing VM $j \in P_i^{\mathrm{e}}$ in PM $i \in W$
q_n	Given parameter indicating requested capacity of new coming VM $n \in N$
c_i^{R}	Given parameter indicating remaining capacity on PM i except for primary allocation of existing VMs
p	Given parameter indicating failure probability of each PM
ϵ	Given small number for probabilistic protection guarantee
x_k^{ij}	Binary variable indicating whether existing VM $j \in P_i^{\mathrm{e}}, i \in W$, is protected by PM $k \in W : k \neq i$
z_k^{in}	Binary variable indicating whether new coming VM $n \in N$ is allocated into PM $i \in W$ and protected by PM $k \in W : k \neq i$
X_i	Random variable indicating whether PM i fails
n_k	Number of PMs, each of which is partially or totally protected by PM k
Γ_k	Number of PMs representing robustness for PM k
c_k^{B}	Required backup capacity on PM k
V	Set of all primary and backup paths
$V^{\mathrm{P}} \subseteq V$	Set of all primary paths
$V^{\mathrm{B}} \subseteq V$	Set of all backup paths
$W^{\mathrm{P}} \subseteq V^{\mathrm{P}}$	Subset of primary paths
$W^{\mathrm{B}} \subseteq V^{\mathrm{B}}$	Subset of backup paths

Notation	Description
v^{P}	Primary path
v^{B}	Backup path
$V_l \subseteq V$	Set of all substrate paths that include substrate link l
$\mathbb{W}_l^{\mathrm{B}}$	Set of all maximal unexclusive subsets on substrate link l
l_{ij}	Given parameter indicating propagation latency between switch $i \in S$ and controller $j \in C$
q_i	Given parameter indicating acceptable unavailability of switch i
b_i	Given parameter indicating propagation latency bound of switch i
p_j	Given parameter indicating failure probability of controller j
c_j	Given parameter indicating capacity of controller j
y_{ij}	Given parameter indicating priority of controller j to become master controller of switch i
x_{ij}	Binary variable indicating whether controller j is assigned to switch i

Abbreviations

Abbreviation	Description
ISP	Internet service provider
InP	infrastructure provider
SP	service provider
NFV	network function virtualization
SDN	software-defined networking
VM	virtual machine
PM	physical machine
CPU	central processing unit
IaaS	infrastructure as a service
PaaS	platform as a service
SaaS	software as a service
IDS	intrusion detection system
NAT	network address translator
LD	load balancer
ETSI	European telecommunications standards institute
NFVI	NFV infrastructure
VNF	virtual network function
MANO	management and network orchestration
VN	virtual network
ILP	integer linear programming
SN	substrate network
WAN	wide-area network
INLP	integer non-linear programming
MILP	mixed integer linear programming

Abbreviation	Description
GLB	general load balancing
SC	Santa Clause
SA	simulated annealing
PP	partition problem
LP	linear programming
LPR	linear programming relaxation
WBM	weighted bipartite b-matching

Chapter 1

Introduction

1.1 Network virtualization

The Internet has achieved great success over the course of past several decades with advancing a wide variety of distributed applications and network technologies. However, its further growth meets the biggest impediment brought by its popularity. With the multi-provider nature, modifying the existing Internet architecture or developing a new one requires consensus among competing stakeholders. Consequently, alterations to the Internet architecture have been limited to simple incremental updates; implementations of new network technologies are difficult [1, 2].

Network virtualization has been introduced as a key role in the next-generation networking paradigm to fend off this ossification. It decouples the traditional Internet service providers (ISPs) into two roles: infrastructure providers (InPs) and service providers (SPs). An InP manages and provides the physical infrastructure to multiple ISPs, each of which provisions end-to-end network services with aggregating resources from multiple InPs. Such an environment releases the inherent limitations of the existing Internet to allow heterogeneous network architectures coexist.

Generally, network virtualization is related to the virtualization of different types of resources in a network. Figure 1.1 shows the basic components in network virtualization. A platform with network virtualization provides virtualized resources of computing, functionality, and networking to users in a

cost-effective and dynamic manner, by leveraging the technologies of computer virtualization, network function virtualization (NFV), and software-defined networking (SDN). This thesis introduces the three key enablers for network virtualization in Sections 1.1.1-1.1.3.

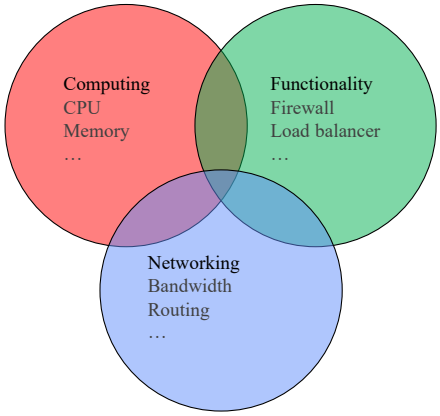


Figure 1.1: Basic components in network virtualization.

1.1.1 Computer virtualization

Computer virtualization, or hardware virtualization, is the process of creating a virtual instance of computer system abstracted from the physical computer or hardware [3]. With computer virtualization, multiple and different operating systems with the form of virtual machines (VM) can run on the same physical machine (PM), which provides computing resources, such as central processing unit (CPU) and memory. It indicates that different users or applications with different needs can share the same hardware. As a result, the physical computing resource is utilized efficiently.

Cloud computing is one of the most successful applications of computer virtualization. It allows global access to shared resources and services that can be quickly provisioned and released with the nominal effort over the Internet. The services offered by cloud providers are typically classified into three categories, which are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [4, 5]. An IaaS provider serves resources that are represented by several types of VM for its customers. Based

on the customers' requirements, the cloud provider allocates different types of VMs to its PMs according to resource allocation policies [6, 7].

1.1.2 Network function virtualization

Network functions, such as firewalls, intrusion detection systems (IDS), network address translators (NAT), and load balancers (LD), are functional blocks within network infrastructures that have well-defined functional behaviors and external interfaces to support network services. Usually, a network function is called as a middlebox in a network. While middleboxes used to be implemented as dedicated physical devices, they now can be implemented as softwares running on commodity servers by adopting the NFV technology to efficiently utilize the hardware resources in networks [8].

As the European telecommunications standards institute (ETSI) introduced, the NFV architecture is composed of the NFV infrastructure (NFVI), virtual network functions (VNF), and the management and network orchestration (MANO). NFVI provides the environment to develop VNFs with combining both physical and virtual resources. The physical resources include physical computing and networking resources that provide processing and transmission capacities for VNFs. The virtual resources are abstractions of physical resources based on a virtualization layer. More specifically, the computing resources can be represented by VMs, while a virtual network (VN) consists of a set of virtual nodes and a set of virtual links [9,10]. A virtual node is a software component, such as an operating system encapsulated in a VM. A virtual link spans over a substrate path with utilizing a certain transmission capacity to connect tow virtual nodes. A VNF is an implementation of network function deployed on a set of virtual resources. For example, a VNF can be composed of one or multiple internal components, each of which can be deployed over a VM. NFV MANO works on all virtualization-specific management tasks, such as the orchestration and lifecycle management for resources supporting VNFs.

1.1.3 Software-defined networking

In traditional networks, as mentioned, network functions are deployed in specific hardwares. Particularly, for switches and routers, algorithms, policies,

and protocols are implemented on specific hardwares to control and forward data flows of dedicated network services. To provide a new network service, the network operator needs to manually updates the switches with the low-level configuration commands to realize the corresponding high-level policies. As a result, severe issues, such as reliability, scalability, and network speed, occurs in the traditional networks.

The SDN paradigm provides a flexible and deployable network architecture by decoupling control and forwarding planes which are highly integrated together in traditional networks. The control plane consists of a set of controllers that decide where and how to forward data in switches placed in the forwarding plane. Typically, a large-scale SDN uses multiple controllers, which are logically centralized but physically distributed in the network [11–13]. With the architecture of SDN, a (logic) single point, i.e., the control plane, controls the whole network, which makes it easy to implement protocols or policies supporting new network services or applications.

1.1.4 Challenges of resource allocation in network virtualization

While network virtualization leads to a more flexible and efficient network, it brings challenges for network management, one of which is the resource allocation in network virtualization. More specifically, compared to traditional networks, there exist more choices or candidates for a decision in resource allocation with network virtualization. Different choices can vary a lot in terms of the resulted performance. How to efficiently allocate the computing, functionality, and networking resources to satisfy different requirements from users or requested services is challenging in network virtualization.

1.2 Reliability issue in network virtualization

As the adaptation of network virtualization in different application archetypes is an increasing trend, the reliability of an environment with network virtualization has became a major concern. It was reported that, from 2010 to 2016, the average cost of a data center outage has increased from $505,502 to

$740,357 [14]. More importantly, customer satisfaction greatly decreases if services are suddenly stop and cannot be recovered promptly with the occurrence of failures.

A network element, such as a PM, network function, or SDN controller, may fail due to several reasons, such as overloading, software crashes, connectivity errors, configuration bugs, and hardware faults [15–17]. Each network element failure can result in several critical issues for the network and the users involved. A PM failure in cloud providers make all VMs hosted by it unavailable for the users. If a middlebox fails and cannot be promptly recovered, the network function in the middlebox becomes unavailable, which interrupts all services using this function. For an SDN, all the switches become unavailable if the corresponding controller fails and no prompt recovery is provided.

Multiple simultaneous failures potentially occur among network elements in network virtualization. For example, several elements can be destroyed at the same time or the failure can occur on the second element before the recovery of the first failed element. Therefore, resource allocation with protection strategies dealing with multiple failures has become an essential requirement for network virtualization.

1.3 Problem statements

This thesis studies five specific problems about reliable resource allocation in network virtualization, each of which includes one or serval questions that have not been addressed and are answered in this thesis.

1.3.1 Probabilistic protection for virtual machines

To protect VMs form multiple PM failures, if the backup resources for VMs in PMs are allocated into their corresponding PMs just as the mirror of the primary resource allocation in the protected PMs, protections with high successful probabilities are provided; especially if a PM hosting backup resources does not fail, the corresponding protected VMs absolutely survive from every random PM failure scenario. For example, we can synchronize each state of an active VM to its standby on the PM providing the protection, or running the

standby in parallel with the active one, to achieve a high degree of fault tolerance [18]. However, this straightforward approach requires double amounts of total capacity in the cloud provider.

Providing a probabilistic protection guarantee allows the primary resources in PMs to share the backup capacity, and consequently the total cost of protection can be reduced benefiting from statistic multiplexing gain. More specifically, the snapshot of a VM is stored and updated to a PM providing the protection. When a protected VM fails, it can be recovered on the corresponding PM based on the snapshot. Since updating the stored snapshot of a VM usually requires much less computing capacity compared to entirely running or recovering the VM, multiple VMs can share the same reserved backup resource [19, 20]. Clearly, a PM may not recover all failed VMs at the same time due to the limited reserved backup capacity, which leads to the failing of protection. A probabilistic protection guarantee is defined as that, for any PM, the probability that the protection provided by the PM fails is guaranteed with no greater than a given number. The resource allocation problem to provide protection against any single PM failure can be formulated as an integer linear programming (ILP) model based on the idea of [21]. It becomes difficult when multiple simultaneous failures occur, each of which has a failure probability.

A question, which has not been addressed, arises: *is there any model that allocates primary and backup resources with providing probabilistic protection against multiple failures?* This work answers this question in Chapter 3.

1.3.2 Probabilistic protection for virtual networks

With network virtualization, services can be represented as VNs embedded in the same substrate network (SN). Each VN consists of a set of virtual nodes and a set of virtual links. A virtual node demands a certain computing capacity, such as CPU and memory capacities, and is hosted on a substrate facility node. A virtual link spans over a substrate path in the SN with utilizing a certain transmission capacity. When a service request arrives, a VN is embedded to the SN based on resource allocation policies [9, 10].

The problem stated in Section 1.3.1 only considers the backup computing

resource allocation for virtual nodes regardless of network aspects, which means that it does not consider the backup resource allocation at a VN level. When some primary facility nodes fail, the embedded virtual nodes are moved to the corresponding backup facility nodes. At the same time, the virtual links connected to the moved virtual nodes need to be remapped to some prepared backup substrate paths, which require a certain allocated backup transmission capacity.

Two questions, which have not been addressed, arise: 1) *how can we jointly consider backup computing and transmission resource allocation against multiple random facility node failures?* 2) *how can we reduce the required backup computing capacity as much as possible after incorporating the backup transmission resource allocation?* This work studies these two questions in Chapter 4.

1.3.3 Backup resource allocation for network functions

The availability of middleboxes is increasingly concerned. Middlebox failures are found to be prevalent in networks and significantly impact hosted services over the past few years [22]. In order to improve the availability of a middlebox, several backup servers can be assigned to protect it, where the information required to recover the middblebox is synchronically updated to each assigned backup server. The degree of synchronization can vary for different application scenarios. The works in [19,20] considered the aspect of limited network resources by allowing backup computing resource sharing, where snapshots of multiple middleboxes can be stored at the same backup server, and a certain number of them can be recovered at the same time by using the reserved computing capacity. Delay may occur when the snapshot of failed middlebox is used to regenerate the missing states [20]. On the contrary, some delay-sensitive applications may require to synchronize each internal state of an active middlebox to its standbys with utilizing a large amount of computing resources, which are not shared among standbys, such that the middlebox can be promptly and correctly recovered by its backup servers when it fails [16, 18, 23]. Note that a backup server in this thesis is not necessarily referred to a physical one and can be a logical server abstracted from a set of resources.

The backup servers are always assumed not to fail, such as the works in [19, 24], and only the failures of functions in middleboxes are considered. In practical NFV environments, backup servers may also fail with some probabilities. The failure of backup server can affect the availability of functions. For example, a function protected by several backup servers, each of which has enough resource to host the function, becomes unavailable when the middlebox and all the corresponding backup servers experience failures simultaneously. Therefore, both failure probabilities of functions and backup servers need to be considered for the availability of middleboxes.

In addition, the importance of different functions can be different in practical applications. For example, in security applications, the achievable level of defense can be increased by guaranteeing a high level of availability for some specific security functions. Similarly, the work in [25] suggested that the necessities of functions included in a service chain are different, where the availabilities of some functions just contribute to improve the quality of service that is completed by other functions. Since the capacity of backup servers is always limited, the importance of functions need to be considered in the assignment of backup servers to efficiently utilize the backup resource.

A question arises: *is there any model that allocates backup resources for functions with considering both failure probabilities of functions and backup servers and the importance of different functions?* This question has not been addressed, and is studied in Chapter 5.

1.3.4 Unavailability-aware shared backup allocation

In literature, the backup allocation problem with the shared protection has been studied from different aspects for middleboxes. The work in [19] considered finding the assignment of backup severs to middleboxes with maximizing the probability for a recovery of all failed middleboxes. The work in [24] studied the maximum number of failed middleboxes that can be fully recovered by a given number of backup servers. However, these works did not explicitly consider the unavailability of middleboxes; consequently, the suggested backup allocations may not be optimal in terms of the middlebox unavailability.

Handling the middlebox unavailability directly in a backup allocation model

requires comprehensive analyses considering heterogeneous procedures. More specifically, each device, middlebox or backup server, in a network may experience heterogeneous procedures that affect the unavailability of middleboxes; for the same procedure, the behaviors may vary for different devices. For example, a device may experience different procedures, waiting or no waiting, to recover its failure depending on the backup allocation; the failure and repair procedures of different devices can be with different characteristics of probability distributions. The work in [26] studied the unavailability with considering heterogeneous procedures for the dedicated protection. Since the shared protection introduces a more complicated mechanism to recover a failed middlebox, developing a backup allocation model being explicitly aware of the middlebox unavailability becomes difficult when the shared protection is adopted.

Two questions arise: 1) *how can we estimate the unavailability of middleboxes with considering the shared protection?* 2) *what is the optimal backup allocation in terms of the unavailability of middleboxes?* The two questions have not been addressed, and are studied in Chapter 6.

1.3.5 Controller assignment for switches

The reliability of control plane in an SDN is a critical issue in practical applications [17]. One approach to protect switches is to assign multiple controllers to each switch, where one of the assigned controllers works as a master controller and others are slave controllers for the switch [27]. At any time, each switch is only managed by its master controller. When the master controller fails, one of the available slave controllers becomes the new master controller to promptly recover the failure.

The propagation latency between switches and controllers is another critical issue for lots of SDNs, such as the wide-area network (WAN) [28]. In WANs, the control reaction of a controller that runs at reasonable speed and stability is bounded by the propagation latency. With enough latency, some tasks may greatly slow down; real-time tasks, such as link layer fault recovery, become infeasible. The work in [29] showed how the number of controllers existing in a network and the placement of these controllers impact the prop-

agation latency between switches and controllers. The propagation latency is analyzed in both average-case and worst-case. The work in [30] presented a controller placement and assignment model to minimize the average-case propagation latency. Given a placement of controllers, the controller assignment problem is formulated as a minimum weight matching problem in [30]. The work in [31] introduced a master controller assignment model to minimize flow setup latency, where propagation latency between master controllers is considered. The master controller assignment problem was formulated as an integer non-linear programming (INLP) problem in [31]. All the works in [29–31] do not consider the survivability issue in an SDN, where each switch only has a master controller without any slave controller.

Given a placement of switches and controllers in an SDN, how to assign master and slave controllers to each switch with considering both survivability and propagation latency is a significant problem. The work in [32] addressed how to assign slave controllers under a single master controller failure; the propagation latency between a switch and any controller assigned to it is limited within a constant value. Multiple failures potentially occur among controllers in an SDN. For example, several controllers may fail simultaneously or the second controller fails before the recovery of the first failed controller is completed. Previously, the works in [33–35] considered multiple controller failures, where a constant number of controllers are assigned to each switch. The worst-case propagation latency among all switches and failure patterns is considered. As the works in [33–35] mainly focused on the aspect of controller placement, several questions remain for controller assignment.

Different switches may require different degrees of availability, and different controllers may have different probabilities to fail. Simply assigning the same number of controllers to each switch may not be efficient in resource utilization. For example, a switch with higher availability requirement needs to be assigned with more backup resources that are related to the number of assigned controllers and their failure probabilities. With the multiple assigned controllers, the order to connect controllers, i.e., the selection for the master controller under each failure pattern from all available assigned controllers, needs to be determined. Typically, in order to obtain the optimal order, it is designed to be determined in the same optimization problem with controller

assignment [33, 35]. Clearly, this approach introduces additional decision variables, and consequently the complexity of problem can be greatly increased.

Two questions, which have not been addressed, arise: 1) *how can we build the assignment model that is aware of the characteristics of different components?* 2) *can we develop a general policy that always yields the optimal order to avoid adding additional decision variables?* This work answers the two questions in Chapter 7.

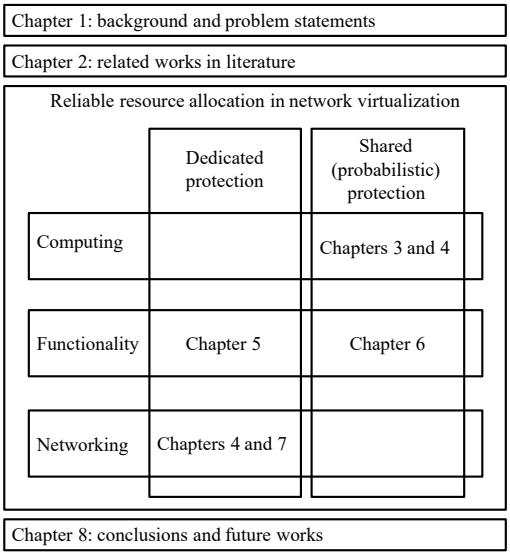


Figure 1.2: Chapter overview of this thesis.

1.4 Overview and contributions of this thesis

Figure 1.2 shows the chapter overview of this thesis. Chapter 2 surveys the related works in literature.

Chapter 3 proposes a primary and backup resource allocation model for VMs with providing a probabilistic protection against multiple PM failures to minimize the required total capacity. By adopting robust optimization with extensive mathematical operations, a mixed integer linear programming (MILP) problem is formulated. The results show that the proposed model saves about one-third of the total capacity in the examined cases; it outperforms the conventional models in terms of both blocking probability and resource utilization.

Chapter 4 incorporates backup transmission resource allocation in the probabilistic protection model for VNs against multiple facility node failures. The backup transmission resource sharing is analyzed in the case of multiple facility node failures to compute the minimum required backup transmission capacity. The results observe that the proposed model reduces the required backup computing capacity compared to the baseline with dedicated protection.

Chapter 5 proposes a backup resource allocation model for middleboxes with considering both failure probabilities of network functions and backup servers. The importance of functions is taken into account. Three heuristic algorithms with polynomial time complexity are developed, where the approximation performances of different heuristics are analyzed with providing several lower and upper bounds. Based on the discussions in numerical results, a network operator can choose an appropriate approach according to the requirements in specific applications.

Chapter 6 proposes an unavailability-aware backup allocation model with the shared protection to minimize the maximum unavailability among functions. An analytical approach based on the queueing theory is developed to compute the middlebox unavailability for a given backup allocation. A simulated annealing (SA) heuristic is introduced to obtain the backup allocation based on the developed analytical approach. The results reveal that the proposed model reduces the maximum unavailability 16% in average compared to the baseline model

Chapter 7 proposes a master and slave controller assignment model against multiple controller failures in software-defined networks. The survivability guarantee of each switch is satisfied by assigning a set of controllers. Given assigned controllers for a switch, the master controller in each failure case is automatically specified based on a low latency first policy, which is proved as the optimal choice for each considered problem. The numerical results observe that, compared to the baseline that introduces decision variables to determine the master controllers, the proposed model obtains the optimal objective value with the computation time about 10^2 times shorter.

Finally, Chapter 8 concludes this thesis and discusses the future works to extend this work.

Chapter 2

Related works

2.1 Resource allocation in computer virtualization

In literature, various studies have been considered for backup resource allocation problems to provide the mirroring protection [18, 36]. Taking this direction, the work in [36] considered providing protection against a single link failure in primary network by developing a dedicated backup network. The work in [18] described an approach to protect VMs from the failure of PMs in terms of any VM-based system. In this thesis, the proposed robust optimization model for VMs [37] and the proposed probabilistic protection model for VNs [38] provide the probabilistic protection guarantee to reduce the required total capacity for primary and backup resource allocation.

Similar to the proposed robust optimization model, several works provided probabilistic protection guarantees for backup resource allocation [19, 39, 40]. The works in [39,40] presented an approach to design a dedicated backup network against random link failures. Robust optimization, where the provided backup capacity in backup links is robust to the uncertainty in primary link failures, is adopted to formulate an ILP problem for backup capacity provisioning. In the proposed model, this work applies the similar technologies to formulate the resource allocation problem with providing the probabilistic protection guarantee against random PM failures as an MILP problem. This work differs from [39, 40] in three major aspects: (i) instead of only backup

resource allocation, this work allocates both primary and backup resources in PMs; (ii) this work allows different PMs to provide backup capacity for different primary VMs which are in the same protected PM; (iii) instead of designing a dedicated backup network, a PM in the proposed model accepts both primary and backup resources. As an approach different from ILP, the work in [19] focused on the heuristic algorithm for backup resource allocation for middleboxes. The work in [38] presented the transmission capacity and flow constraints for backup resource allocation of virtual networks based on the results of [37].

Robust optimization has been applied to network flow problems [41–43]. The work in [41] formulated an MILP for maximizing the amount of admissible VPN traffic. The traffic demand model adopted in [41] is called the hose model, where the uncertainty of the traffic-demand matrix is resisted by setting traffic upper bounds outgoing from ingress nodes and incoming to egress nodes in the network. The work in [42] considered a two-phase routing to achieve a traffic-oblivious routing in IP-over-optical networks. The work in [43] applied robust optimization for energy saving against link traffic uncertainty.

Some studies focused on the reliability of a cloud provider, where the backup resource allocation for high survivability is not considered [44,45]. The works in [44, 45] presented reliable resource allocation models for distributed clouds from a networking aspect. In their models, a request of VMs is allocated in several selected PMs which are located in the different nodes of cloud network and are considered as a star topology where the center PM is assumed not to fail.

2.2 Resource allocation in NFV

Since middlebox failures are found to be prevalent in networks and significantly impact hosted services [22], some works designed the frameworks for fault-tolerant middleboxes [16, 46, 47]. Taking this direction, the work in [46] adopted flow-level checkpoints to provide the synchronization per flow. The work in [16] presented a framework where the traffic packets and associated determinant information of a stateful middlebox are stored in its backup servers, which can recover the primary middlebox with reintroducing the traffic packets

to the failed network function by using the stored copy of state. Based on [16], the work in [47] introduced an approach with avoiding using the snapshots of entire system to reduce the latency overhead.

Addition to the framework designing, similar to the works in [48–50], several works focused on the basic backup resource allocation problem for middleboxes or VMs from different aspects [19, 24]. The work in [19] provided several properties of an optimal assignment of backups servers to functions to maximize the probability for a full recovery or to maximize the expected number of functions that can be recovered simultaneously. The work in [24] designed the recovery schemes, which guarantee a full recovery for any small subset of failed functions, based on the analyses of machine graph. The proposed backup allocation model for middleboxes differ from the above works by considering both failure probabilities of backup servers and functions and aiming at different objective.

Some studies addressed the availability issue for service function chain (SFC) instead of focusing on each function [26, 51–56]. The works in [51, 52] presented algorithms mapping SFCs with minimizing the amounts of on-site and off-site backups required for satisfying the availability requirements. The work in [26] presented an approach to provision availability of SFC with considering heterogeneous failure processes; an optimization problem was formulated to minimize the required backup resources with satisfying a certain degree of availability requirement for an SFC request. The work in [53] jointly considered the availability and delay constraints in the backup resource allocation problem for SFC with minimizing the required bandwidth resource, and the further works in [54, 55] considered backup resource sharing and multipath routing. The work in [56] considered the software characteristics of virtual functions, where the backup sharing for basic and duty resources is clarified.

Compared to the works in [26,51–56], the work in [50]introduces a different way to estimate the middlebox availability. In [26, 51–56], the availability of a middlebox required in a SFC is calculated with considering the given failure probabilities or availabilities of related devices. This approach dose not include some factors, such as the recovery procedure for a failed middlbox on its backup server and the action after the shared protection fails, which also affect the availability. In this thesis, the availability of each middlebox

is estimated based on the queueing theory with comprehensively considering the heterogeneous failure, repair, recovery, and waiting procedures of functions and backup servers.

This work is also related to the algorithm designing for the general load balancing (GLB) problem [57–61] and the Santa Clause (SC) problem [62, 63]. The works in [58, 59] provided a greedy algorithm with an upper bound of $\frac{4}{3}$ of optimum for the GLB problem; algorithms with stronger approximation guarantees for this problem were presented in [61] based on the dual approximation technique. The work in [63] studied the SC problem with providing an algorithm based on the LPR approach, which has an upper bound similar to the one proved in Section 5.3.3 for considered problem. The work in [62] introduced another approximation algorithm for the SC problem by configuring a different LP problem and showed the integrality gap of the configuration LP problem. In this thesis, the backup resource allocation problem for middleboxes is equivalent to maximizing the minimum, which is opposite with the objective in the GLB problem; it also varies from the GLB problem and the SC problem by considering two conditions, which are that each function has a corresponding initial workload and that one backup server can be assigned to several different functions.

The queueing theory has been wildly studied for different network problems [64–68]. The work in [64] presented a fiber span power management scheme by using the dummy wavelength signals to shorten the lightpath provisioning and releasing times; the system blocking probability and the request waiting time are estimated through a queueing model. The work in [65] developed the analytical models based the queuing theory to predict the average packet delay and packet loss probability of flows over software-defined networks. The works in [66–68] studied the VNF deployment problem, where each VNF instance is modeled as an M/M/1 or M/M/c queueing model to process the traffic flows. In this work for the unavailability-aware backup allocation model, the whole system including functions and backup servers with the heterogeneous procedures is not equivalent to the traditional M/M/1 or M/M/c model; this work provides the comprehensive analyses in the developed queueing approach.

2.3 Resource allocation in SDN

Since the control plane of an SDN frequently suffers from failures in practical applications, several works implemented the replication mechanisms to improve the survivability of control plane [69, 70]. The work in [69] developed functional implementations for SDNs with passive and active replications; it showed that the passive and active replications are suitable for a less intrusive approach and a delay-sensitive scenario, respectively. The work in [70] focused on actively replicating controllers, where a prototype with consistent state among distributed controllers is implemented based on Ryu controllers and OpenReplica. Instead of replication mechanisms, the work in [71] studies a basic problem of the assignment between controllers and switches.

This work is related to controller placement problems [29, 30, 33–35, 72–76]. Taking this direction, one type of work studies how to place controllers with the objectives focusing on the propagation latency; the survivability issue is not considered, such as [29, 30]. Another type of work studies resilience aware controller placement. The works in [72, 73] addressed placing controllers to satisfy the network reliability requirement against network link failures regardless of the propagation latency. The works in [33–35, 74–76] considered placing controllers with different objectives against multiple controller failures. Usually, the assignment of controller in a placement problem is intuitively obtained when the placement is determined. For example, the work in [29] assumed that each switch connects the nearest controller; the works in [74–76] considered assigning a constant number of nearest controllers to each switch. Some placement problems, such as [30, 33–35], determine the controller assignment. Compared to the assignment aspect studied in their works, this work has three main differences: i) instead of assigning the same number of controllers to each switch, this work considers the characteristics of different switches and controllers; ii) this work adopts a policy-based approach and develops an optimal policy to determine the master controllers; iii) instead of considering the worst-case propagation latency among all failure patterns, this work introduces expected values to reflect the probabilities of different failure patterns. Note that this work can be extended to include the controller placement procedure.

Similar to this work, several works studied the controller assignment prob-

lem for the given placement of switches and controllers [31, 77–83]. The work in [31] developed a low-complexity algorithm for the master controller assignment problem. The work in [77] focused on the efficient controller assignment in a network virtualization environment. The works in [78] and [79] studied the controller assignment against traffic dynamics. The assignment of slave controllers is not considered in [31, 77–79]. Some works addressed assigning slave controllers against the link failures instead of the controller failures. For example, the work in [80] considered the survivability of a domain which is defined as the average probability that the connection between the controller and each switch in this domain does not fail. The works in [81–83] considered the controller failure in the controller assignment problem. The work in [81] introduced a multiple controller mapping model against a single controller failure. The work in [82] extended the work in [32] by introducing a dynamic slave controller assignment scheme, which determines slave controllers after each controller fails. The work in [83] explored the modes of existing SDN switches for the efficient recovery; a model was presented to select a set of offline switches to be configured with the legacy routing mode, and to remap the remaining offline switches with the SDN mode to active controllers, under controller failures. Different from the above works, this work deals with multiple controller failures by proactively assigning master and slave controllers before any failure occurs for a prompt recovery.

This work is also related to the algorithm designing for optimization problems with submodular objective functions [84–86]. The work in [84] showed that a greedy algorithm achieves a $(1 - 1/e)$-approximation for maximization problems with submodular objective functions and uniform matroid constraints. The works in [85] extended the results in [84] by considering the case of intersection of p matroids. The work in [86] adopted the pipage rounding technique to develop a greedy algorithm with $(1 - 1/e)$-approximation under a general matroid constraint.

Chapter 3

Robust optimization model for virtual machine allocation

This thesis proposes a primary and backup resource allocation model with providing a probabilistic protection against multiple PM failures to minimize the required total capacity. A part of the work in this chapter was presented in [37]. A probabilistic protection guarantee is introduced to assure the recovery of failed PMs with a certain probability. The proposed model adopts an efficient resource allocation policy, where the capacity of a PM is allowed to accept both primary and backup resources. In addition, suppressing capacity fragmentation is considered for further efficient resource utilization. Allowing backup capacity sharing with probabilistic protection guarantee and efficient resource utilization leads to cost reduction of protection.

To develop the proposed model, inspired by [39, 40], this work considers robust optimization to provision backup capacity. Robust optimization deals with the problem where the uncertainty of the problem is resisted with a certain measure of robustness. In the case considered in this thesis, uncertainty can be represented as the backup capacity required to provide protection. By adopting robust optimization with extensive mathematical operations, an MILP problem is formulated. This work proves that the primary and backup resource allocation problem is NP-hard. SA is introduced to solve the same optimization problem.

This work considers both static and dynamic scenarios for performance

evaluation. This work compares the proposed model with two conventional models to express the significance of the proposed model in terms of blocking probability and resource utilization in a cloud provider. In the static scenario, this work observes that the proposed model saves one-third of the total capacity by providing the probabilistic protection; the results from the SA heuristic are equal to the optimal values in the examined cases. In the dynamic scenario, the results reveal that the proposed model outperforms the conventional models in terms of both blocking probability and resource utilization.

The rest of this chapter is organized as follows. Section 3.1 presents the proposed robust optimization model. The proof of NP-hardness is provided in Section 3.2. Section 3.3 introduces the SA heuristic. The dynamic scenario is described in Section 3.4. Section 3.5 evaluates the performance of the proposed model. Section 3.6 summaries this chapter.

3.1 Optimization model

Consider a cloud provider that consists of a set of PMs, which is denoted by W. This thesis studies the computing resource allocation for VMs with assuming that the sufficient and reliable network resources are provided.

3.1.1 Protection of virtual machines in cloud provider

The capacity of a PM is used to accept both primary and backup resources. On one hand, the cloud provider offers resources used for VMs to customers based on the PMs. On the other hand, excepting for offering capacity for primary resource allocation, each PM is able to provide protections for VMs with allocating backup resources. More specifically, the latest snapshot of a running VM is periodically updated to its corresponding PM that provides the protection. When PMs experience failures, the corresponding survived PMs recover the failed VMs based on the snapshots and take over the workloads. Since updating the snapshot usually requires much less computing capacity than entirely running or recovering a VM, the reserved backup capacity can be shared among different VMs. Of course, if multiple PMs where hosted VMs share the same backup resource fail simultaneously, only some of the failed

VMs can be recovered due to the limited capacity, which leads to the failing of protection provided by this PM. Note that a PM can just be protected by other PMs instead of itself. Figure 3.1 presents an example that shows a cloud provider consisting of six PMs.

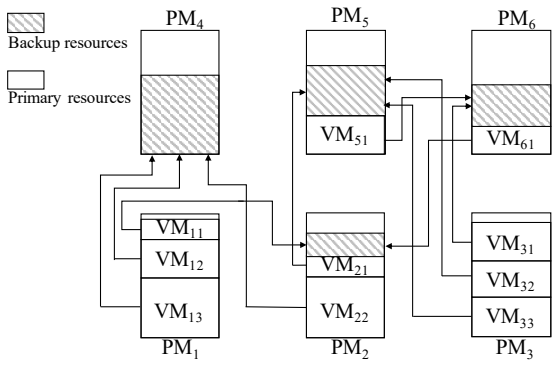


Figure 3.1: Example of cloud provider, where each PM is used to accept both primary and backup resources.

3.1.2 Primary and backup resource allocation model

This work builds an optimization model to minimize the total capacity required for the primary and backup resource allocation. Table 3.1 summarizes the frequently used notations.

The set of VMs existing in PM $i \in W$ is denoted by P_i^{e}. The capacity of a VM typically includes multiple types of resources, such as CPU and memory. This thesis assumes that the capacity has a single type of resource to simplify the discussion. The capacity of VM $j \in P_i^{\mathrm{e}}$ in PM $i \in W$ is represented by r_{ij}^{e}. When a request for a set of VMs comes to the cloud provider from customers, primary and backup resource allocation is performed for the requested VMs. At the same time, backup resources for existing VMs are re-allocated to minimize the total capacity required in the cloud provider. To avoid any service disruption, the primary resources for existing VMs are not re-allocated.

N denotes a set of requested VMs. The capacity of VM $n \in N$ is represented by q_n. $z_k^{in}, k \in W, i \in W : i \neq k, n \in N$, denotes a binary decision variable; z_k^{in}

Table 3.1: List of frequently used notations in Chapter 3.

	Notations	Meaning
Given parameters	r_{ij}^{e}	Requested capacity of existing VM $j \in P_i^{\mathrm{e}}$ in PM $i \in W$
	q_n	Requested capacity of new coming VM $n \in N$
	c_i^{R}	Remaining capacity on PM i except for primary allocation of existing VMs
	p	Failure probability of each PM
	ϵ	Given small number for probabilistic protection guarantee
Variables	x_k^{ij}	Binary variable indicating whether existing VM $j \in P_i^{\mathrm{e}}, i \in W$, is protected by PM $k \in W : k \neq i$
	z_k^{in}	Binary variable indicating whether new coming VM $n \in N$ is allocated into PM $i \in W$ and protected by PM $k \in W : k \neq i$
	X_i	Random variable indicating whether PM i fails
	n_k	Number of PMs, each of which is partially or totally protected by PM k
	Γ_k	Number of PMs representing robustness for PM k
	c_k^{B}	Required backup capacity on PM k

is set to one if VM $n \in N$ is allocated into PM $i \in W$ and protected by PM $k \in W$ and zero otherwise.

$x_k^{ij}, k \in W, i \in W : i \neq k, j \in P_i^{\mathrm{e}}$, denotes a binary decision variable; x_k^{ij} is set to one if VM $j \in P_i^{\mathrm{e}}$ in PM $i \in W$ is protected by PM $k \in W$ and zero otherwise. c_k^{B} denotes the required capacity of PM $k \in W$ for backup resource allocation. The available capacity in PM $i \in W$ for primary and backup resource allocation is denoted by c_i^{R}, which is the remaining capacity except for the primary allocation of existing VMs. p_i denotes the failure probability of PM $i \in W$.

S is a set of triplets (i, j, k), each of which does not allow $x_k^{ij} = 1$. For example, a PM is not allowed to be protected by itself, so $x_k^{ij} = 0$ when $k = i$; or if the connection between resource $(i, j), i \in W, j \in P_i$ and $k \in W$ is not satisfied with the management requirement, $x_k^{ij} = 1$ is not allowed.

Obviously, S is significant for the total required capacity for resource allocation. For example, if the different VMs in one PM is not allowed to be protected by different PMs due to some management restrictions, much higher capacity is required in PMs compared with the situation that one PM is allowed to be protected by different PMs for different resources.

This work assumes $p_i = p, \forall i \in W$, to simplify the discussion. The assumption that all the failure probabilities for PMs are the same p can be relaxed

by using the similar approach in [40]. X_i denotes an independent, identically distributed Bernoulli random variable with parameter p, indicating whether PM $i \in W$ fails.

$$\min \quad \sum_{k \in W} c_k^{\mathrm{B}} \tag{3.1a}$$

$$s.t. \quad \sum_{k \in W: k \neq i} x_k^{ij} = 1, \forall i \in W, j \in P_i^{\mathrm{e}} \tag{3.1b}$$

$$\sum_{k \in W} \sum_{i \in W: i \neq k} z_k^{in} = 1, \forall n \in N \tag{3.1c}$$

$$P(X_k = 1) \times$$
$$P\left[\sum_{i \in W: i \neq k} X_i \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n\right) > 0\right] +$$
$$P(X_k = 0) \times$$
$$P\left[\sum_{i \in W: i \neq k} X_i \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n\right) > c_k^{\mathrm{B}}\right] \leq \epsilon,$$
$$\forall k \in W \tag{3.1d}$$

$$c_i^{\mathrm{B}} + \sum_{k \in W: k \neq i} \sum_{n \in N} z_k^{in} q_n \leq c_i^{\mathrm{R}}, \forall i \in W \tag{3.1e}$$

$$x_k^{ij} = 0, \forall (i, j, k) \in S \tag{3.1f}$$

$$x_k^{ij} \in \{0, 1\}, \forall k \in W, i \in W : i \neq k, j \in P_i^{\mathrm{e}} \tag{3.1g}$$

$$z_k^{in} \in \{0, 1\}, \forall k \in W, i \in W : i \neq k, n \in N. \tag{3.1h}$$

Equation (3.1a) minimizes the total required capacity for backup resource allocation. Equation (4.12c) ensures that each existing VM in a PM is protected by another PM. Equation (3.1c) ensures that each requested VM is allocated into a PM and protected by another PM. Equation (3.1d) is the probabilistic protection guarantee, which indicates that the resources in PMs must be provisioned to be protected by their corresponding PMs, where the probability that the protection fails is not greater than ϵ. The first term of left hand side in (3.1d) indicates the probability that the PM providing protection fails and at least one protected PM fails. The second term of left hand side in (3.1d) indicates the probability that the PM providing protection does not fail and

the protection fails due to insufficient backup capacity. Since x_k^{ij} is a binary decision variable, resource j in $i \in W$ is not divided into more than one PM. Equation (4.12e) restricts that the total required capacity for primary and backup resource allocation does not exceed c_i^{R}.

Note that this model can be widely used. Specially, when N is an empty set, this model is used for backup resource allocation for P_i^{e}. When P_i^{e} is an empty set for all $i \in W$, this model is for primary and backup resource allocation for N.

3.1.3 Mixed integer linear programming problem

This work shows how to compute the probability that the protection provided by a PM fails. More specifically, in order to solve the primary and backup resource allocation model by using optimization solvers [87, 88], this work transforms (3.1a)-(3.1h) to an MILP problem by expressing the probabilistic protection guarantee in (3.1d) as a linear form.

This work defines

$$n_k = \sum_{i \in W: i \neq k} \left\{ \min \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} + \sum_{n \in N} z_k^{in}, 1 \right) \right\}, \tag{3.2}$$

where n_k is the number of PMs, each of which is partially or totally protected

by PM $k \in W$. Equation (3.1d) is expressed by,

$$
\begin{aligned}
& P(X_k = 1) \times \\
& P\left[\sum_{i \in W: i \neq k} X_i \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n\right) > 0\right] + \\
& P(X_k = 0) \times \\
& P\left[\sum_{i \in W: i \neq k} X_i \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n\right) > c_k^{\mathrm{B}}\right] \\
= & p\{1 - (1-p)^{n_k}\} + \\
& (1-p) P\left[\sum_{i \in W: i \neq k} X_i \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n\right) > c_k^{\mathrm{B}}\right] \leq \epsilon, \\
& \forall k \in W. \qquad (3.3)
\end{aligned}
$$

Let

$$
\epsilon' = \frac{\epsilon - p\{1 - (1-p)^{n_k}\}}{1-p}, \forall k \in W. \qquad (3.4)
$$

Equation (3.3) is transformed to,

$$
P\left[\sum_{i \in W: i \neq k} X_i \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n\right) > c_k^{\mathrm{B}}\right] \leq \epsilon', \forall k \in W. \qquad (3.5)
$$

Therefore, provisioning a certain capacity c_k^{B} in PM $k \in W$ for backup resource allocation to meet the probabilistic protection guarantee in (3.1d) is equivalent to satisfying (3.5) as long as $\epsilon \geq p\{1 - (1-p)^{n_k}\}$. Actually, for PM $k \in W$, when the approach mirroring primary resources for backup resource allocation is adopted, the probability that the protection provided by PM k fails is $p\{1 - (1-p)^{n_k}\} = p^2$ with $n_k = 1$, which is the lower bound of ϵ in the proposed model.

In order to express (3.1d) or (3.5) in a linear form, this work starts by considering the unit-capacity case where $\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n$ is 1 or 0. In Section 3.1.3, this work removes the assumption to consider a general-capacity case, where the idea in robust optimization is used to provision backup capacity

to satisfy (3.5). After adopting the dual theorem and several mathematical transformations, (3.1a)-(3.1h) are finally transformed to an MILP problem in Section 3.1.3.

Unit-capacity case

This work assumes that $\sum_{j\in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n\in N} z_k^{in} q_n$ is 1 or 0, when x_k^{ij} and z_k^{in} are obtained as solutions. Equation (3.5) is expressed by,

$$P\left[\sum_{i\in W: i\neq k} X_i \left(\sum_{j\in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n\in N} z_k^{in} q_n\right) > c_k^{\mathrm{B}}\right] \tag{3.6a}$$

$$= \sum_{y=\lfloor c_k^{\mathrm{B}}\rfloor+1}^{n_k} \binom{n_k}{y} p^y (1-p)^{n_k-y} \leq \epsilon', \forall k \in W. \tag{3.6b}$$

Let $G(n_k, p, \epsilon)$ be the minimum integer value of c_k^{B} that satisfies (3.6). This work denotes $\Gamma_k = G(n_k, p, \epsilon)$; p and ϵ are given parameters, which are omitted for simplicity, but Γ_k depends on p and ϵ.

Obviously, Γ_k represents the number of PM failures protected against. In the unit-capacity case, for PM $k \in W$, the probabilistic protection guarantee (3.5) is satisfied by allocating backup capacity for any Γ_k PMs among all the protected PMs, and $c_k^{\mathrm{B}} = \Gamma_k$.

General-capacity case

This work considers a general-capacity of r_{ij}^{e} and q_n by removing the assumption of $\sum_{j\in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n\in N} z_k^{in} q_n = 1$ or 0. Since the capacity of primary resources varies in different protected PMs in the general-capacity case, simply allocating backup capacity for any Γ_k PMs among all the protected PMs may not satisfy (3.5). This work adopts the idea in robust optimization to provision backup capacity to satisfy (3.5) and further transform it in a linear form.

Robust optimization deals with the problem where the uncertainty of problem is resisted with a certain measure of robustness. The work in [89] introduced a robust approach to formulate the problems with uncertain data as linear optimization problems with providing a controllable degree of robustness. A prespecified optimization parameter Γ of coefficients changes is introduced

in the approach, where sufficient (robust) capacity is provided to guarantee that the solution is feasible if no more than Γ uncertain coefficients change, and the solution is proved to be feasible with a certain probability when there are more than Γ uncertain coefficients changing. The works in [39, 40] applied the robust optimization results of [89] to design a dedicated backup network against random link failures.

In this problem, the number of failed PMs follows the binomial distribution whose probability mass function is expressed in (3.6). However, it is uncertain which PM fails, which leads to an uncertainty in the required backup capacity in each PM of the general-capacity case. This work applies the similar technologies with [39, 40, 89] in this problem to provide a certain measure of robustness in the required backup capacity to resist the uncertainty.

Let L_k be a set of PMs $i \in W : i \neq k$, each of which is partially or totally protected by another PM $k \in W$, i.e., $L_k = \{i | \sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} + \sum_{n \in N} z_k^{in} \geq 1\}$ and the size of L_k is n_k. Given n_k, this work computes the corresponding Γ_k by using (3.6). Let S_k be a set of Γ_k PMs in L_k with the largest capacities protected by PM $k \in W$. For any $i \in S_k$, we have

$$\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n \geq$$
$$\sum_{j \in P_{i'}^{\mathrm{e}}} x_k^{i'j} r_{i'j}^{\mathrm{e}} + \sum_{n \in N} z_k^{i'n} q_n, \forall i' \in L_k \backslash S_k, k \in W. \tag{3.7}$$

The backup capacity required to protect against any Γ_k PMs in L_k is given by,

$$c_k^{\mathrm{B}} = \sum_{i \in S_k} \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n \right) \geq \tag{3.8a}$$

$$\max_{S_k | S_k \subseteq L_k, |S_k| = \Gamma_k} \left\{ \sum_{i \in S_k} \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n \right) \right\}. \tag{3.8b}$$

If no more than Γ_k PMs among the n_k protected PMs fail simultaneously, the backup capacity allocated by (3.8b) is sufficient. Therefore, the probability in the left hand of (3.5), which is the probability of insufficient backup capacity, can be upper-bounded by the probability that more than Γ_k PMs

fail simultaneously, which is not greater than ϵ'. As a result, (3.5) is satisfied for the general-capacity case by adopting the above robust optimization technologies.

Therefore, for a general capacity of r_{ij}^{e} and q_n, (3.1a)-(3.1h) are transformed to the following optimization problem by replacing (3.1d) with (3.9c).

$$\min \quad \sum_{k\in W} c_k^{\mathrm{B}} \tag{3.9a}$$

$$s.t. \quad (4.12c), (3.1c), (4.12e), (3.1f), (3.1g), (3.1h) \tag{3.9b}$$

$$c_k^{\mathrm{B}} \geq \max_{S_k | S_k \subseteq L_k, |S_k| = \Gamma_k} \left\{ \sum_{i\in S_k} \left(\sum_{j\in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n\in N} z_k^{in} q_n \right) \right\}, \quad \forall k \in W. \tag{3.9c}$$

Further transformations

This work adopts the duality technique, which is a similar approach in [40], to express the right hand side of (3.9c) to a linear form. Consider the following primal problem. For each $k \in W$, x_k^{ij}, z_k^{in} and Γ_k are fixed and $w_k^i, k \in W, i \in W : i \neq k$ is introduced as a decision variable.

$$\max \quad \sum_{i\in W: i\neq k} \left(\sum_{j\in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n\in N} z_k^{in} q_n \right) w_k^i \tag{3.10a}$$

$$s.t. \quad \sum_{i\in W: i\neq k} w_k^i \leq \Gamma_k \tag{3.10b}$$

$$0 \leq w_k^i \leq 1, \forall i \in W : i \neq k. \tag{3.10c}$$

The dual of (3.10a)-(3.10c) is formulated by,

$$\min \quad \nu_k \Gamma_k + \sum_{i\in W} \theta_k^i \tag{3.11a}$$

$$s.t. \quad \nu_k + \theta_k^i \geq \sum_{j\in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n\in N} z_k^{in} q_n, \quad \forall i \in W : i \neq k \tag{3.11b}$$

$$\nu_k \geq 0 \tag{3.11c}$$

$$\theta_k^i \geq 0, \forall i \in W : i \neq k, \tag{3.11d}$$

where ν_k and θ_k^i are introduced as decision variables in the dual formulation.

By setting w_k^i in (3.10a)-(3.10c) as a real variable in a range of [0, 1], there is no duality gap between (3.10a)-(3.10c) and (3.11a)-(3.11d). The duality theorem guarantees that the optimal value of (3.10a) in the primal problem is equivalent to that of (3.11a) in the dual problem. The linear programming problem maximizes the objective value of (3.10a) by choosing the Γ_k PMs with the largest $\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n$ for $w_k^i = 1$ and setting other values of w_k^i to zero. As a result, the optimal value of (3.10a) is equivalent to the right hand side of (3.9c). Thus, the right hand side of (3.9c) is equivalent to the optimal value of (3.11a) in the dual problem.

Equations (3.9a)-(3.9c) are transformed by the following optimization problem.

$$\min \quad \sum_{k \in W} c_k^{\mathrm{B}} \tag{3.12a}$$

$$s.t. \quad (4.12\mathrm{c}), (3.1\mathrm{c}), (4.12\mathrm{e}) - (3.1\mathrm{h}) \tag{3.12b}$$

$$c_k^{\mathrm{B}} \geq \nu_k \Gamma_k + \sum_{i \in W} \theta_k^i, \forall k \in W \tag{3.12c}$$

$$\nu_k + \theta_k^i \geq \sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n,$$

$$\forall k \in W, i \in W : i \neq k \tag{3.12d}$$

$$\nu_k \geq 0, \forall k \in W \tag{3.12e}$$

$$\theta_k^i \geq 0, \forall k \in W, i \in W : i \neq k. \tag{3.12f}$$

n_k is a function of decision variables, and affects Γ_k. As Γ_k cannot be computed analytically, this work prepares a table whose mth entry denotes $\Gamma(m)$, which is equal to $G(m, p, \epsilon)$. $\Gamma(m)$ is computed numerically in advance.

To compute n_k, this work introduces a binary decision variable, v_k^m, where v_k^m is set to one if $n_k = m$, and zero otherwise. We have the following constraint given by,

$$\sum_{m=0}^{|W|-1} v_k^m = 1, \forall k \in W. \tag{3.13}$$

v_k^m is set to 1 for only one value of m for each $k \in W$. By using (3.2), we have,

$$\sum_{i \in W: i \neq k} \left\{ \min \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n, 1 \right) \right\} = \sum_{m=0}^{|W|-1} m \cdot v_k^m, \forall k \in W. \tag{3.14}$$

To express $\min \left(\sum_{j \in P_i^{\mathrm{e}}} x_k^{ij} r_{ij}^{\mathrm{e}} + \sum_{n \in N} z_k^{in} q_n, 1 \right)$ in a linear form, let α_k^i be a binary decision variable. Equation (3.14) is transformed into the following constraints, which is used in the minimization problem.

$$\sum_{i \in W: i \neq k} \alpha_k^i \leq \sum_{m=0}^{|W|-1} m \cdot v_k^m, \forall k \in W. \tag{3.15}$$

$$x_k^{ij} \leq \alpha_k^i, \forall k \in W, i \in W : i \neq k, j \in P_i \tag{3.16}$$

$$z_k^{in} \leq \alpha_k^i, \forall k \in W, i \in W : i \neq k, n \in N \tag{3.17}$$

$$\alpha_k^i \leq \sum_{j \in P_i} x_k^{ij} + \sum_{n \in N} z_k^{in}, \forall k \in W, i \in W : i \neq k. \tag{3.18}$$

Equations (3.16) and (3.17) force $\alpha_k^i = 1$ if at least one $j \in P_i$ satisfying $x_k^{ij} = 1$ or at least one $n \in N$ satisfying $z_k^{in} = 1$ exits for each $k \in W, i \in W : i \neq k$. Equation (3.18) forces $\alpha_k^i = 0$ if $\sum_{j \in P_i} x_k^{ij} + \sum_{n \in N} z_k^{in} = 0$ for each $k \in W, i \in W : i \neq k$. Equations (3.15)-(3.18) give m PMs protected by another PM k. Γ_k is expressed by,

$$\Gamma_k = \sum_{m=0}^{|W|-1} \Gamma(m) v_k^m, \forall k \in W. \tag{3.19}$$

Equation (3.12c) is transformed into

$$c_k^{\mathrm{B}} \geq \sum_{m=0}^{|W|-1} \Gamma(m) \nu_k v_k^m + \sum_{i \in W: i \neq k} \theta_k^i, \forall k \in W. \tag{3.20}$$

As the product of $\nu_k v_k^m$ is not linear, this work expresses $\nu_k v_k^m$ with an introduced non-negative decision variable, y_k^m, that satisfies the following constraints. This work defines $W^+ = \{0, 1, \cdots, |W| - 1\}$.

$$y_k^m \geq \nu_k + K(v_k^m - 1), \forall k \in W, m \in W^+ \tag{3.21}$$

$$y_k^m \leq K v_k^m, \forall k \in W, m \in W^+ \tag{3.22}$$

$$y_k^m \geq 0, \forall k \in W, m \in W^+ \tag{3.23}$$

K is a sufficiently large number that satisfies $K \geq \sum_{i \in W} \sum_{j \in P_i^e} r_{ij}^W + \sum_{n \in N} q_n$. If $v_k^m = 0$, y_k^m is forced to zero by (3.22). If $v_k^m = 1$, y_k^m is forced to ν_k in the optimization problem by (3.21).

Equations (3.12a)-(3.12f) are transformed into the following MILP problem.

$$\min \quad \sum_{k \in W} c_k^B \tag{3.24a}$$

$$s.t. \quad (4.12c), (3.1c), (4.12e) - (3.1h), (3.12d) - (3.12f), (3.13)$$

$$(3.15) - (3.18), (3.21) - (3.23) \tag{3.24b}$$

$$c_k^B \geq \sum_{m=0}^{|W|-1} \Gamma(m) y_k^m + \sum_{i \in W: i \neq k} \theta_k^i, \forall k \in W \tag{3.24c}$$

$$v_k^m \in \{0, 1\}, \forall k \in W, m \in W^+ \tag{3.24d}$$

$$\alpha_k^i \in \{0, 1\}, \forall k \in W, i \in W : i \neq k. \tag{3.24e}$$

To reduce the computation time of solving the above MILP problem, this work utilizes a characteristic of $\Gamma(m)$, which is a non-decreasing step function on m. This work classifies the values of m into several classes based on the table whose mth entry denotes $\Gamma(m)$. Let M' be the set of these classes. $D_{m'}$ denotes a set of m, which are with the same value of $\Gamma(m)$, in class $m' \in M'$. The maximum value of m in $D_{m'}$ is represented by $L(m')$. Equations (3.24a)-(3.24e) are transformed into the following MILP problem.

$$\min \quad \sum_{k \in W} c_k^B \tag{3.25a}$$

$$s.t. \quad (4.12c), (3.1c), (4.12e) - (3.1h), (3.12d) - (3.12f), (3.16) - (3.18),$$

$$(3.24e) \tag{3.25b}$$

$$c_k^B \geq \sum_{m' \in M'} \Gamma(L(m')) y_k^{m'} + \sum_{i \in W: i \neq k} \theta_k^i, \forall k \in W \tag{3.25c}$$

$$\sum_{m' \in M'} v_k^{m'} = 1, \forall k \in W \tag{3.25d}$$

$$\sum_{i \in W: i \neq k} \alpha_k^i \leq \sum_{m' \in M'} L(m') \cdot v_k^{m'}, \forall k \in W \tag{3.25e}$$

$$y_k^{m'} \geq \nu_k + K(v_k^{m'} - 1), \forall k \in W, m' \in M' \tag{3.25f}$$
$$y_k^{m'} \leq K v_k^{m'}, \forall k \in W, m' \in M' \tag{3.25g}$$
$$v_k^{m'} \in \{0, 1\}, \forall k \in W, m' \in M' \tag{3.25h}$$
$$y_k^{m'} \geq 0, \forall k \in W, m' \in M'. \tag{3.25i}$$

3.1.4 Extended model suppressing capacity fragmentation

The above primary and backup resource allocation model only considers the minimization of required backup capacity regardless of the optimization in capacity fragmentation. Different solutions of primary and backup resource allocation with the same required backup capacity may cause different degrees of capacity fragmentation in the cloud provider, where the heavier capacity fragmentation is, the less efficient resource utilization is in the cloud provider in terms of dynamic scenarios.

In the extended model, the number of PMs hosting the primary and backup resources after resource allocation is used to measure the capacity fragmentation in the cloud provider. $d_i, i \in W$, denotes a binary given parameter; d_i equals to one when PM $i \in W$ is used to host the primary resource of any existing VM and zero otherwise. $d'_i, i \in W$, denotes a binary decision variable; d'_i is set to one if PM $i \in W$ is used to host any resource after resource allocation and zero otherwise. The primary and backup resource allocation model to suppress capacity fragmentation is represented as an MILP problem by extending the MILP problem in (3.25a)-(3.25i) as follows.

$$\min \quad \sum_{k \in W} c_k^{\mathrm{B}} + \gamma \sum_{i \in W} d'_i \tag{3.26a}$$
$$s.t. \quad (4.12c), (3.1c), (4.12e) - (3.1h), (3.12d) - (3.12f), (3.16) - (3.18),$$
$$(3.24e), (3.25c) - (3.25i) \tag{3.26b}$$
$$d'_i \geq d_i, \forall i \in W \tag{3.26c}$$
$$d'_i \geq \sum_{k \in W: k \neq i} z_k^{in}, \forall i \in W, n \in N \tag{3.26d}$$
$$d'_k \geq \sum_{i \in W: i \neq k} z_k^{in}, \forall k \in W, n \in N \tag{3.26e}$$

$$d'_k \geq x_k^{ij}, \forall k \in W, i \in W : i \neq k, j \in P_i \tag{3.26f}$$

$$d'_i \in \{0,1\}, \forall i \in W, n \in N. \tag{3.26g}$$

The objective given by (3.26a) is still to minimize the required backup capacity. The second term of (3.26a), which sums up the number of PMs used for hosting any resource after the primary and backup resource allocation, is used to suppress capacity fragmentation. Unless specifically stated, γ is set to small enough in order to not impair the first term of (3.26a), or γ is satisfied with $\gamma < \frac{\delta C}{|W|}$, where δC is a minimum incremental value of C_k^B. This provides the primary and backup resource allocation that has a less degree of capacity fragmentation among solutions that have the same value for the first term of (3.26a). Equations (3.26c)-(3.26f) express that d'_i is set to one if PM $i \in W$ is used to host any resource after resource allocation.

3.2 NP-hardness

From the primary and backup resource allocation (PBRA) problem, this work defines the decision version of PBRA problem as below:

Problem Given a set of existing VMs, P_i^e, for each PM $i \in W$ and a set of requested VMs N, is it possible to allocate the primary resources of N and the backup resources of P_i^e and N, which satisfy the probabilistic protection guarantee, with the total required capacity no more than c?

Theorem 1 *The PBRA decision problem is NP-hard.*

Proof : This work shows that the partition problem (PP), which is a known NP-complete problem [90], is reducible to the PBRA decision problem. PB is defined as: is it possible to partition a given set G of positive integers into two subsets G_1 and G_2 such that the sum of the numbers in G_1 equals that in G_2?

This work constructs an instance of the PBRA decision problem from any instance of PP. An instance of PP consists a multiset G of positive integers and the value of positive integer $g \in G$ is represented by I_g. An instance of the PBRA decision problem is constructed with the following algorithm, which performs in a polynomial time of $O(|G|)$.

1) This work considers a cloud provider which consists of three PMs, PM_1, PM_2 and PM_3. Only PM_1 is hosting a set of existing VMs P_1^e with $|P_1^e| = |G|$. For each positive integer $g \in G$, there is a corresponding VM $j \in P_1^e$ with the capacity of $r_{1j}^e = I_g$. No VM exists in PM_2 and PM_3.

2) The maximum capacities of PM_1, PM_2 and PM_3 are set to $\sum_{j \in P_1^e} r_{1j}^e$, $\frac{\sum_{j \in P_1^e} r_{1j}^e}{2}$ and $\frac{\sum_{j \in P_1^e} r_{1j}^e}{2}$, respectively. Therefore, the remaining capacities of PM_1, PM_2 and PM_3 are 0, $\frac{\sum_{j \in P_1^e} r_{1j}^e}{2}$ and $\frac{\sum_{j \in P_1^e} r_{1j}^e}{2}$, respectively.

3) Set N as an empty set and consequently this model is used for backup resource allocation for P_1^e.

4) Set $\epsilon = 0$, which means that 100% protection is required against any random failure.

5) Set $c = \sum_{g \in G} I_g = \sum_{j \in P_1^e} r_{1j}^e$.

Consider that a PP instance is a Yes instance. G is able to be partitioned into two subsets G_1 and G_2, and both sums of the numbers in the two subsets are $\frac{\sum_{g \in G} I_g}{2}$. Define a PBRA instance from the PP instance by using the above described algorithm. In the PBRA instance, for each existing VM $j \in P_1^e$, consider a corresponding backup resource $b \in B$ with capacity $f_b = r_{1j}^e$. The set of backup resources B is able to be partitioned into two subsets B_1 and B_2, which refer to G_1 and G_2, respectively, and both total capacities of the backup resources in the two subsets are $\frac{\sum_{j \in P_1^e} r_{1j}^e}{2}$. By allocating B_1 and B_2 into PM_2 and PM_3, respectively, 100% protection is provided for the primary resources of P_1^e. As a result, it is possible to allocate the backup resources of P_1^e, which satisfy the probabilistic protection guarantee, into the PMs with the total required backup capacity no more than $c = \sum_{j \in P_1^e} r_{1j}^e$. Therefore, the PBRA instance is a Yes instance.

Conversely, consider a PBRA instance is a Yes instance. In the Yes PBRA instance, in order to provide 100% protection, for each existing VM $j \in P_1^e$, a corresponding backup resource $b \in B$ with capacity $f_b \geq r_{1j}^e$ has to be allocated into PM_2 and PM_3. As the total required backup capacity is less or equal to $c = \sum_{j \in P_1^e} r_{1j}^e$, the capacity of each backup resource $b \in B$ is $f_b = r_{1j}^e$. Since the

remaining capacities of PM_2 and PM_3 are $\frac{\sum_{j \in P_1^e} r_{1j}^e}{2}$ and $\frac{\sum_{j \in P_1^e} r_{1j}^e}{2}$, respectively, the set of backup resources $\boldsymbol{B}$ is able to be partitioned into two subsets $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$, both total capacities of the backup resources in the two subsets are $\frac{\sum_{j \in P_1^e} r_{1j}^e}{2}$. Referring to the partition of $\boldsymbol{B}$, we are able to partition $\boldsymbol{G}$ into two subsets $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ such that the two sums of numbers in the two subsets equal to each other. Therefore, if the PBRA instance is a Yes instance, then the PP instance is a Yes instance.

Note that the above described algorithm transforms any PP instance into a PBRA instance in a polynomial time. This work confirmed that if a PP instance is a Yes instance, then the corresponding PBRA instance is a Yes instance, and vice versa. This proves that PP, a known NP-complete problem, is polynomial time reducible to the PBRA decision problem. Thus, the PBRA decision problem is NP-hard. □

3.3 Simulated annealing

When the problem size is small, such as the values of $|W|$ and $|N|$ are small, the MILP problem introduced in (3.26a)-(3.26g) can be solved in a practical time, but for large one, it cannot due to an increase of computation time. SA is introduced to solve the same optimization problem in (3.26a)-(3.26g) when the problem size is large.

SA is a heuristic method inspired by physics for solving an optimization problem [91]. The annealing in physics means the process of cooling an alloy with high temperature down slowly. This procedure can result in low energy allocation for the atoms. From an algorithmic point of view, the annealing procedure is represented by a "temperature" variable $\boldsymbol{T}$ which is set to be high at the beginning and gradually decreases with each iteration. A random solution is set and its related cost is computed at the beginning of algorithm. In each iteration, a random change is applied to an existing solution to generate a new solution and a new cost is also computed. If the new cost is less than the existing one, the new solution is accepted. In addition, a worse solution, which has a higher cost than the existing one, can also be accepted with a probability depending on the two costs and $\boldsymbol{T}$, which avoids a local minimum

solution.

For the primary and backup resource allocation problem in (3.26a)-(3.26g), when SA is applied, a solution can be represented by two vectors combined by x_k^{ij} and $z_k^{in}, k \in W, i \in W : i \neq k, j \in P_i, n \in N$, respectively. Furthermore, when each solution is generated, (4.12c) must be satisfied to ensure that each existing VM is protected by a PM; (3.1c) must be satisfied to ensure that each requested VM is allocated into a PM and protected by another PM. The required backup capacity for each PM is computed by (3.8). If the solution dose not satisfy the capacity constraint of (4.12e), its cost is set to a large enough constant, such as the one greater than $\sum_{i \in W} \sum_{j \in P_i^{\mathrm{e}}} r_{ij}^{\mathrm{e}} + \sum_{n \in N} q_n + |W|$. Otherwise, the cost is obtained by computing (3.26a), where the first part is obtained by summing up all the required capacity over each PM and the second part is the product of γ and the number of PMs hosting the primary and backup resources after resource allocation. In each iteration, a new solution is generated by randomly choosing an existing VM and making it be protected by a random PM or randomly choosing a requested VM and making it be allocated into a random PM and protected by another random PM. Then, the cost of new solution $c_{\mathrm{Total}}^{\mathrm{B}'}$ is recomputed and compared with current cost $c_{\mathrm{Total}}^{\mathrm{B}}$. If the new cost is less than the existing one, the new solution is unconditionally accepted. Otherwise, it is accepted with a probability q, which is given by $q = e^{(\frac{c_{\mathrm{Total}}^{\mathrm{B}} - c_{\mathrm{Total}}^{\mathrm{B}'}}{T})}$.

At the beginning of algorithm, the probability that a worse solution is accepted is high, since T is set to be high initially. This attempt can prevent the final solution from local minima. As T gradually decreases by $T' = \rho T$ with $0 < \rho < 1$ in each iteration, the probability accepting a worse solution becomes less and less and finally close to 0. Then, a final solution containing the specific assignment and required backup capacity of each PM is obtained at the end of algorithm.

Since the complexities for computing (3.8) and (3.26a) are $O(|W|(\sum_{i \in W} |P_i| + |N|)^2)$ and $O(|W|)$, respectively, the computational time complexity of each iteration in SA is $O(|W|(\sum_{i \in W} |P_i| + |N|)^2)$, which is polynomial. Note that there is a tradeoff between the accuracy of solution and the total computation time, when SA is adopted. By increasing the total computation time, or choosing a larger value of ρ, we can obtain a more accurate solution.

3.4 Dynamic scenario

This work describes dynamic scenarios, where both situations of requested VMs arriving and existing VMs releasing are considered, to evaluate the performance of proposed model.

3.4.1 Overview

To avoid any service disruption, the primary resources for existing VMs are always not re-allocated. To maintain the minimum reserved backup capacity in dynamic scenarios, one approach is to re-optimize the complete backup resource allocation at each request arriving and releasing. However, this approach with backup resource re-allocation may not be efficient in terms of other aspects, such as computational complexity, network bandwidth, and quality of service. This work considers a dynamic approach, where the backup resources of existing VMs are not re-allocated when the primary and backup resources of requested VMs are allocated and those of terminated VMs are released; a re-optimization procedure with backup resource re-allocation performs at a certain interval to reduce the reserved backup capacity.

3.4.2 Dynamic approach

At any moment, the information of primary and backup resource allocation of existing VMs can be collected to be used as the given parameters for each computation. Let binary given parameter $u_k^{ij}, k \in W, i \in W : i \neq k, j \in P_i^{\mathrm{e}}$, represent the backup resource allocation for existing VMs; u_k^{ij} equals one if the existing VM $j \in P_i^{\mathrm{e}}$ in PM $i \in W$ is protected by PM $k \in W$ and zero otherwise. The overall process of dynamic approach is presented in Fig. 3.2.

When a request for a set of VMs arrives at the cloud provider, it is enqueued into a finite queue to wait the answer whether the cloud provider accepts it. If there is no available space in the queue, the request will be rejected immediately [92]. This arriving procedure is executed in parallel with the flowchart.

The cloud provider answers each request by computing the primary and backup resource allocation for the requested VMs in a first come first served

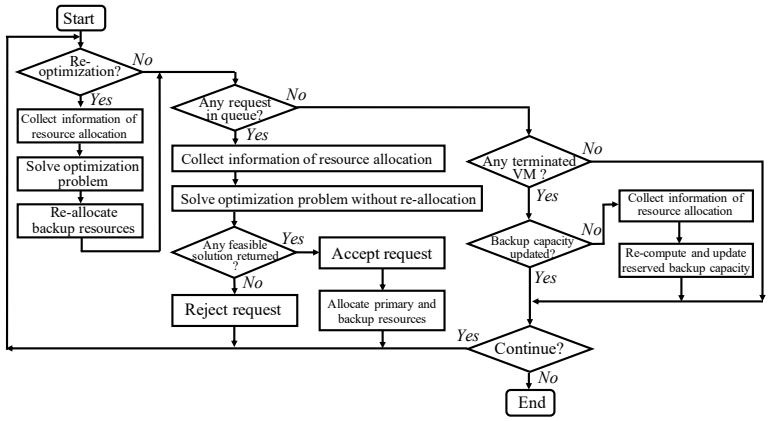


Figure 3.2: Flowchart of dynamic approach.

manner. In each computation, the primary and backup resource allocation of requested VMs and the updated required backup capacity are obtained by solving the MILP problem in (3.26a)-(3.26g) or the same optimization problem using a heuristic method, where decision variable $x_k^{ij}, k \in W, i \in W : i \neq k, j \in P_i^{\mathrm{e}}$, is replaced by given parameter u_k^{ij}. If there is no feasible solution returned within a certain admissible computation time, which means that there is no enough capacity to host the primary and backup resources of requested VMs, the cloud provider rejects the request. Otherwise, the cloud provider accepts the request and allocates the primary and backup resources referring to the obtained solution. In practical applications, the admissible computation time should be set to satisfy the response time requirement [93], which may vary for different types of services. For example, an interactive service can have a strict response time requirement of at most a few seconds [94].

The re-optimization procedure is performed at a certain interval, which is specified by the cloud provider, to reduce the reserved backup capacity. In each re-optimization procedure, the original optimization problem is re-solved based on the primary resource allocation of existing VMs, and then the backup resources of existing VMs are re-allocated. If there is any request waiting in the queue during the re-optimization procedure, the request is processed after the completion of re-optimization procedure. If the interval of two re-optimization procedures is short, the backup resource allocation can keep optimal and the reserved backup capacity can always be minimum. However, performing the

re-optimization procedure frequently may increase operational overhead. A cloud provider can select an appropriate interval and start time for the re-optimization procedure according to the practical environments.

When some VMs are terminated, the occupied primary physical resources and other resources used for backup purpose, such as the stored copies of image files, for the terminated VMs are released immediately; this releasing procedure is executed in parallel with the flowchart. If at least one of the procedures of re-optimization and computing the answer for one request runs after the termination of VMs, the reserved backup capacity of existing VMs is updated by solving the optimization problem during the procedure. Otherwise, it needs to be updated by re-computing (3.8).

3.4.3 Performance metrics

Let K represent a set of requests incoming to the cloud provider. $|K|$ denotes the total number of requests in K. The cloud provider rejects the request directly when the queue is full. In addition, the request is rejected after enqueuing in the queue if there is no sufficient capacity remaining in the cloud provider. The number of rejected requests in K is represented by k_r. Similar to [92], blocking probability P_{b}, which is a probability that a request incoming to the cloud provider is rejected, can be a performance metric for a cloud provider; it is defined by $P_{\mathrm{b}} = \frac{k_r}{|K|}$.

For a cloud provider, the more capacity is used for primary resource allocation, the more profit is gained. Resource utilization, u, which is a ratio of the average capacity used for hosting the primary resources to the total capacity in the cloud provider, is introduced to measure the efficiency of the resource allocation model.

3.5 Numerical results

To observe the effect of the proposed model, this work compares it with two conventional models, which are a non-sharing protection model, named NSP, and a specialized backup PM model, named SBPM. In NSP, backup resource for each primary VM is allocated into another PM just as the mirror of the pri-

mary resource allocation and backup capacity sharing is not allowed. Through modifying the MILP problem in (3.26a)-(3.26g), the MILP problem of NSP is obtained, where only suppressing capacity fragmentation is left in the objective function. In SBPM, the PMs in one cloud provider are divided into two types, which are backup PMs and primary PMs; primary PMs are only used for primary resource allocation and backup PMs are only responsible to take the workloads from the primary PMs when any failure occurs in order to provide probabilistic protection. α represents a fixed parameter in SBPM, which is the ratio of the number of primary PMs to the total number of PMs in a cloud provider. This work extends the MILP problem presented in [37] by incorporating primary resource allocation and suppressing capacity fragmentation to the MILP problem for SBPM.

In the numerical analysis, the capacity of CPU is considered to be backed up. The maximum capacity of each PM in the cloud provider is considered as 1500 million instructions per second (MIPS). The workloads are clarified into three types of VMs, which are small, medium, and large. The capacities required for one small, medium and large VMs are 250 MIPS, 500 MIPS and 750 MIPS, respectively [95,96]. The MILP problems are solved by the IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer with version 12.7.1 [88], using Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory.

3.5.1 For small-size problems

This work considers a cloud provider with ten PMs. Since the problem size is small, this work solves the MILP problem introduced in (3.26a)-(3.26g) to compute the optimal primary and backup resource allocation in terms of static and dynamic scenarios.

Static scenario

This work sets N as an empty set, and consequently the proposed model is used for backup resource allocation for P_i^{e}. Figure 3.3 shows the backup resource allocation for P_i^{e} with $p = 0.025$ and $\epsilon = 0.01$. The backup resources are allocated into two PMs and the required backup capacities of two PMs are 500 MIPS and 1000 MIPS, respectively. Figure 3.3 observes that the proposed

model only needs to allocate 30% of the capacity that is needed for NSP. This is because providing the probabilistic protection guarantee allows backup capacity to be shared with each other. Consequently, the capacity is utilized efficiently and the cost of protection is greatly reduced. More importantly, this work observes that the PMs in the proposed model are used to accept both primary and backup resources. As a result, the capacity of PMs in one cloud provider is utilized more flexibly in the proposed model than that of SBPM.

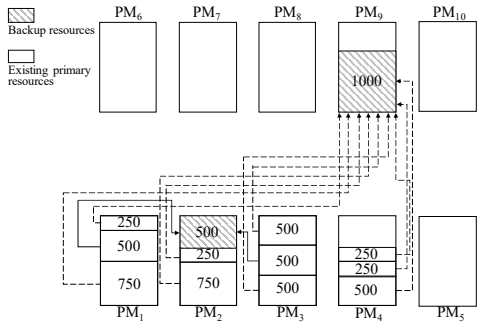


Figure 3.3: Backup resource allocation for P_i^e.

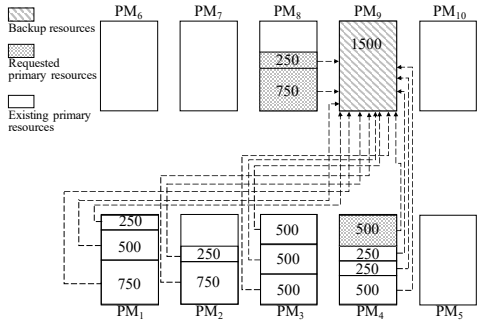


Figure 3.4: Primary resource allocation for N and backup resource allocation for P_i^e and N.

Then, N is set as a set of three requested VMs and the required capacity of each requested VM is 250 MIPS, 500 MIPS and 750 MIPS, respectively. Figure 3.4 shows the primary resource allocation for N and the backup resource allocation for P_i^e and N with $p = 0.025$ and $\epsilon = 0.01$. The proposed model only allocates 23% of primary capacity for backup. Table 3.2 shows the optimal required backup capacity obtained by solving the MILP problem and the results by using the SA heuristic for different values of p when $\epsilon =$

0.01. This work observes that, as p increases, the required backup capacity increases. This is because as the failure probability of PM decreases, more primary resources share backup capacity, which results in less capacity required for backup resource allocation. Contrarily, with higher p, little backup capacity is not sufficient to provide high probabilistic protection. As a result, the required backup capacity is higher when PMs are more unreliable. Furthermore, this work observes that the results from the SA heuristic are equal to the optimal values for different values of p in the examined cases.

Table 3.2: Required backup capacity by solving MILP problem and results from SA heuristic for different values of p. $\epsilon = 0.01$ in each case.

p	0.025	0.03	0.035	0.0425	0.05
MILP	1500	1750	2250	3000	3250
SA	1500	1750	2250	3000	3250

Dynamic scenario

This work applies the proposed model, NSP, and SBPM to the dynamic scenario described in Section 3.4. Usually, the re-optimization procedure may be conducted in free time, such as during the system maintenance period, and the interval of adjacent re-optimization procedures can be much longer than that of adjacent requests and the service time for a request. This work focuses on the dynamic scenario between two adjacent re-optimization procedures to observe the benefit brought by the proposed model.

This work considers that the number of requested VMs of each request is uniformly distributed over the range of $[1, 5]$ and the type of each requested VM is randomly selected from the three types. The requests arrive at the cloud provider based on a Poisson process with arrival rate λ per unit time; the service time for requests follows an exponential distribution with the average of μ^{-1} unit time; the performances of different models are evaluated in terms of the blocking probability of P_{b} and the resource utilization of u [92,97]. This work assumes that the computation time for a request is much smaller than the interval of adjacent requests. It indicates that the effects of computation time on the performances of cloud provider can be ignored. Based on the

assumption, the computation time for a request is considered as zero in the simulation for a dynamic scenario. This work sets the queue size large enough, which means that, in terms of blocking probability, this work only focuses on the case that there is no sufficient capacity remaining in the cloud provider. Simulation results are obtained with a 95% confidence interval that is not greater than 5% of the reported average results.

Figure 3.5 shows the blocking probability depending on parameter γ for different arrival rates λ when the proposed model is adopted in the cloud provider; p, ϵ, and μ are set to 0.015, 0.012, and 1, respectively. In Fig. 3.5, there are three cases according to the value of γ. Cases 1, 2, and 3 are $\gamma = 0$, $\gamma = 0.1$, and $\gamma = 1000000$, respectively, where 0.1 is less than the value of $\frac{\delta C}{|W|}$. This work observes that, for each λ, the blocking probability in case 2 is less than that in case 1, where capacity fragmentation is not suppressed. This is because the objective in case 2 is set to minimize the required backup capacity and suppress the capacity fragmentation with less weight at the same time. As a result, the capacity is utilized more efficiently and less blocking probabilities are obtained. In addition, this work observes that the lowest blocking probabilities for all the λ are obtained in case 2 instead of case 3, where minimizing the required number of PMs hosting the primary resources is the main objective. It indicates that minimizing the required backup capacity is more significant for reducing the blocking probability than minimizing the required number of PMs hosting the primary resources. In the following results presented in Figs. 3.6-3.8, this work sets $\gamma = 0.1$.

Figure 3.6 shows the relationship between blocking probability and parameter α for different PM failure probabilities p when SBPM is adopted in the cloud provider; ϵ, λ, and μ are set to 0.005, 4, and 1, respectively. This work observes that the blocking probability decreases, as α increases from 0 to a certain point. This is because that more capacity is used for primary resource allocation as well as the backup capacity is maintained sufficiently. As a result, the blocking probability, which is caused by insufficient capacity to host the primary resources, decreases. After one point, as α increases, the capacity used for backup resource allocation becomes insufficient. Consequently, the blocking probability turns to increase and backs to 1 when α becomes 1. Therefore, there is a point of α where the blocking probability becomes min-

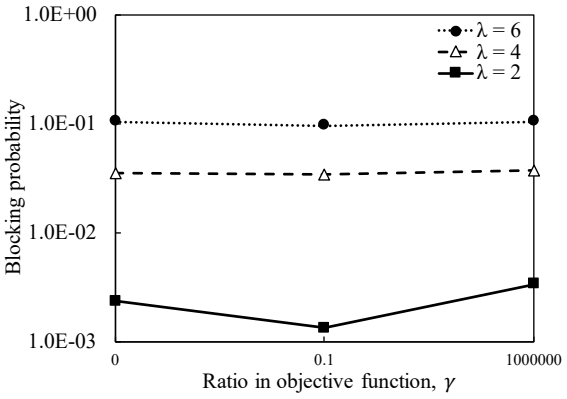


Figure 3.5: Dependency of blocking probability on γ for different λ.

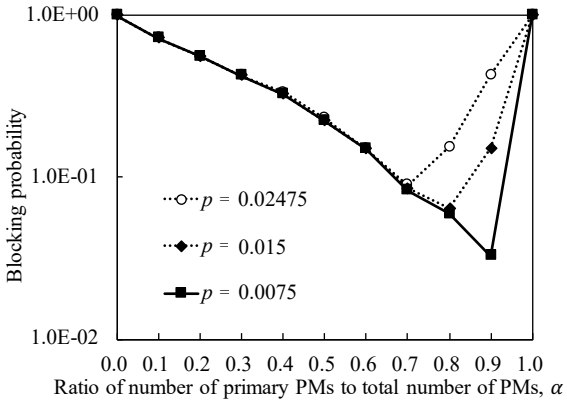


Figure 3.6: Dependency of blocking probability on α for for different p in SBPM.

imum. This work observes that the point of α values to obtain the minimum blocking probability varies for different p values. Actually, it is also affected by multiple parameters, such as ϵ, λ, and N. As a result, it is difficult to find a suitable fixed α, which leads to the minimum blocking probability under all the situations, for one cloud provider.

Figure 3.7 shows the comparison of blocking probabilities of the cloud provider using different resource allocation models obtained for different λ values when $\mu = 1$; p and ϵ are set as 0.015 and 0.005, respectively. This work notices that the blocking probabilities for all the models increase as λ increases. For SBPM, the best performance is obtained when α is fixed to 0.8.

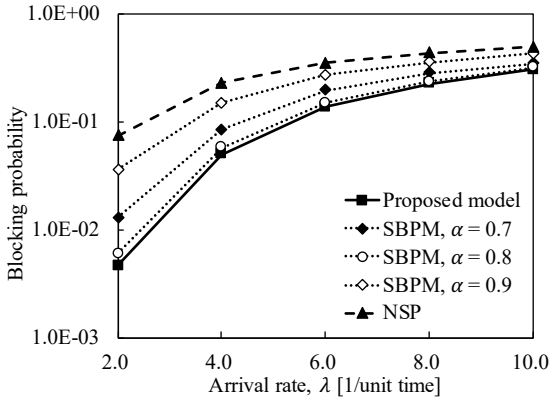


Figure 3.7: Comparison of blocking probabilities using different models for different λ when $\mu = 1$.

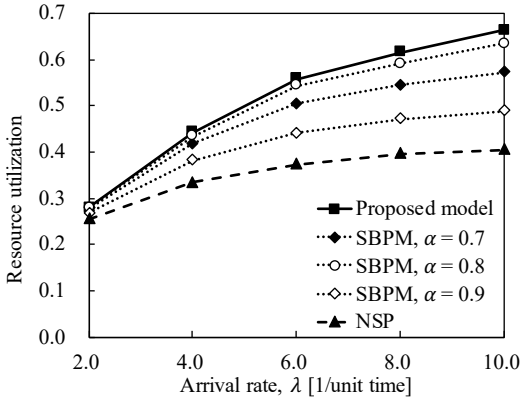


Figure 3.8: Comparison of resource utilization using different models for different λ when $\mu = 1$.

This work observes that the proposed model always outperforms all the conventional models in terms of blocking probability. This is because, compared with NSP, the proposed model allows primary resources in different PMs to share the common backup capacity by providing the probabilistic protection guarantee, and consequently the total capacity required for each primary and backup resource allocation can be reduced benefiting from statistic multiplexing gain, which results in the reduction in the blocking probability. Compared with SBPM, all the PMs are used to accept both primary and backup resources in the proposed model, which leads to more efficient utilization of the capacity of PMs. As a result, the blocking performance of the cloud provider is improved by adopting the proposed model; specifically, compared to SBPM, the blocking probability is reduced about 33% in average by the proposed model. This work observes that the improvement of proposed model on the blocking probability decreases as the arrive rate increases. This is because the blocking probabilities of all models gradually converge to one as the arrive rate increases, which decreases the differences between them.

Figure 3.8 compares the resource utilization in the cloud provider using the proposed model and considered conventional models for different λ values when $\mu = 1$; p and ϵ are set as 0.015 and 0.005, respectively. This work observes that the resource utilization for all the models increase as λ increases. For SBPM, the best performance is obtained when α is fixed to 0.8. The capacity in the cloud provider is always utilized more efficiently by adopting the proposed model rather the conventional models. This is because that each PM in the cloud provider is used for accepting primary resources and providing probabilistic protection for other PMs at the same time. As a result, the proposed model provides more efficient utilization of resources in the cloud provider compared with the considered conventional models.

3.5.2 For large-size problems

Consider the primary and backup resource allocation for a cloud provider comprising 1000 PMs. The MILP problem introduced in (3.26a)-(3.26g) cannot be solved in a practical time for such a large size, but this work solves the same optimization problem by SA.

Static scenario

This work considers a static scenario where 400 PMs are hosting at least one VM and the remaining 600 PMs are not. For each PM of the 400 PMs, the number of existing VMs is uniformly distributed over the range of $[1, 6]$, where the type of each existing VM is randomly set with the limitation that the total capacity of existing VMs does not exceed the maximum capacity of PM. N is considered as a set of ten requested VMs and the type of each requested VM is randomly selected from the three types. Let $\beta \in (0, 1]$ denote a ratio of the required backup capacity in a feasible solution to the total primary capacity of existing and requested VMs.

Figure 3.9 shows the relationship between β and the admissible computation time [s] when the SA heuristic with different settings of ρ is applied to solve the proposed model, where the initial and final values of T are set to 10^5 and 10^{-5}, respectively, $p = 0.0015$, and $\epsilon = 0.0012$. This work obtains that the SA with different ρ takes about 6.2 [s] in average to obtain the first feasible solution, where the required backup capacity is about 0.57 times in average of the primary capacity. It indicates that if the admissible computation time is not greater than 6.2 [s], the proposed model can address the static scenario with 1000 PMs. This work observes that the computation times to obtain the first feasible solution and to obtain the optimal solution in the scenario with 1000 PMs are about 10^3 and at least 10^3 times of those in the scenario with ten PMs, respectively. Generally, β decreases as the admissible computation time increases for each value of ρ. When the admissible computation time is small, the value of β in the SA heuristic with smaller value of ρ is smaller, but the heuristic is terminated earlier with greater final value of β. It indicates that we can set ρ based on the admissible computation time. For example, if the admissible computation time is within 10 [s], ρ is set to 0.90; if the admissible computation time is up to 10^4 [s], such as re-optimization in free time, ρ is set to 0.99.

Dynamic scenario

This work considers that the number of requested VMs of each request is uniformly distributed over the range of $[1, 10]$, where the type of each requested

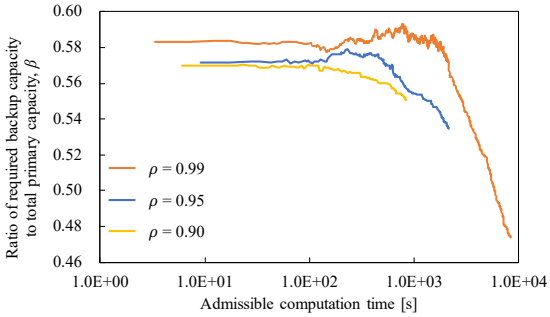


Figure 3.9: Relationship between required backup capacity in feasible solution and admissible computation time [s].

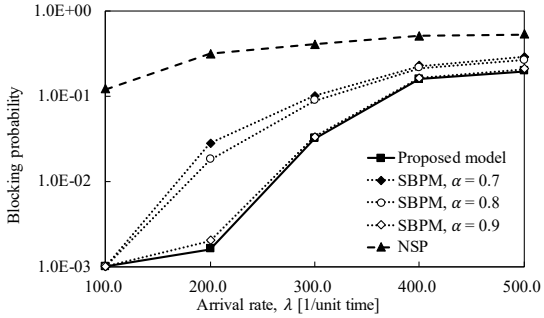


Figure 3.10: Comparison of blocking probabilities of large problem using different models for different λ when $\mu = 1$.

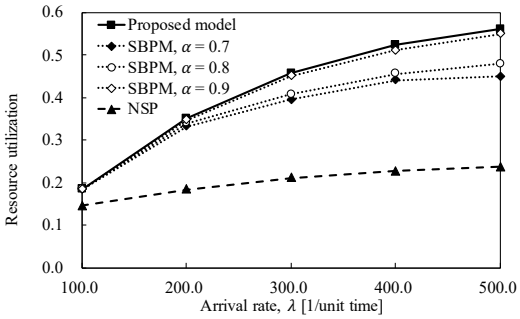


Figure 3.11: Comparison of resource utilization in large problem using different approaches for different λ when $\mu = 1$.

VM is randomly selected from the three types.

Figures 3.10 and 3.11 show the blocking probabilities of the cloud provider versus λ and the resource utilization in the cloud provider versus λ, respec-

tively, when different resource allocation models are adopted; p, ϵ, and μ are set as 0.0015, 0.0012, and 1, respectively. In Figs. 3.10 and 3.11, this work has the observations similar to those of Figs. 3.7 and 3.8, respectively. Specifically, compared to SBPM, the blocking probability is reduced about 31% in average by the proposed model. In summary, this work observes that the proposed model outperforms the conventional models in terms of both blocking probability and resource utilization.

3.6 Chapter summary

This chapter proposed a primary and backup resource allocation model that provides a probabilistic protection guarantee for VMs against multiple PM failures to minimize the required total capacity. A PM in the cloud provider is used to accept both primary and backup resources. Considering the probabilistic protection guarantee in a general-capacity cloud provider leads to a nonlinear programing problem for primary and backup resource allocation against multiple failures. By adopting robust optimization with extensive mathematical operations, this work formulated the primary and backup resource allocation problem as an MILP problem, where capacity fragmentation is suppressed. This work proved that the primary and backup resource allocation problem is NP-hard by showing that the partition problem is reducible to it. For the problem with a large size, we introduced the SA heuristic. Numerical results observed that about one-third of the total capacity is saved in the examined cases by adopting the proposed model. In addition, this work evaluated different models in dynamic scenarios, where both situations of the requested VMs arriving and the existing VMs releasing are considered. The results revealed that the proposed model outperforms the conventional models in terms of both blocking probability and resource utilization.

Chapter 4

Probabilistic protection model for virtual networks

This thesis proposes a backup computing and transmission resource allocation model for VNs with providing the probabilistic protection against multiple facility node failures to minimize the required backup computing capacity. A part of the work in this chapter was presented in [38]. Backup computing resource is allocated such that the probability that the protection from a backup facility node fails due to insufficient reserved backup computing capacity is within a given value. This work describes constraints to provision backup paths, which show the interactions between backup computing and transmission resource allocation. The routing of backup paths is determined when the backup computing resource allocation is determined. The required backup transmission capacity can affect the required backup computing capacity. Considering that different substrate paths which are not used simultaneously can share the same reserved backup transmission resource, this work analyzes backup transmission resource sharing in the case of multiple facility node failures.

The proposed model with non backup transmission resource sharing is formulated as an MILP problem. This work introduces a heuristic algorithm to solve the problem with different degrees of backup transmission resource sharing. Especially, several techniques based on graph theory are developed to handle the problem with full backup transmission resource sharing. This work

compares the proposed model to a baseline with the dedicated protection for computing resource. The results observe that the proposed model outperforms the baseline in terms of both feasibility and required backup computing capacity. Furthermore, the proposed model with full backup transmission sharing, which minimizes the required backup transmission capacity, leads to the least required backup computing capacity. The proposed model with limited backup transmission sharing saves the computation time with a slight performance degradation.

The rest of this chapter is organized as follows. Section 4.1 presents the proposed optimization model. Section 4.2 analyzes the backup transmission resource sharing. Section 4.3 introduces the heuristic algorithm. The performances of different models are evaluated in Section 4.4. Section 4.5 summaries this chapter.

4.1 Model and problem definition

4.1.1 Virtual networks in substrate network

An SN, which consists of a set of substrate facility nodes and a set of substrate links, is an infrastructure to support services. Let undirected graph $G^{\mathrm{S}}(N^{\mathrm{S}}, L^{\mathrm{S}})$ represent the SN, where N^{S} and L^{S} represent the sets of substrate facility nodes and substrate links, respectively. Each substrate facility node is connected to the SN via a dedicated router or switch, which is assumed not to fail. Substrate facility node $n \in N^{\mathrm{S}}$ provides computing resources with the maximum capacity of c_n. The set of neighbors of substrate facility node $n \in N^{\mathrm{S}}$ is denoted by $A(n)$. The transmission capacity of substrate link $l \in L^{\mathrm{S}}$, which is assumed not to fail, is represented by b_l.

A service is represented as a VN, which consists of sets of virtual nodes and links. The VN is embedded into the SN through mapping each virtual node and virtual link to a substrate facility node and a substrate path, respectively. Several VNs can be embedded into the same SN, each of which utilizes a certain computing and transmission resources. Let M denote a set of VNs embedded in the SN $G^{\mathrm{S}}(N^{\mathrm{S}}, L^{\mathrm{S}})$. VN $m \in M$ is represented by undirected graph $G_m^{\mathrm{D}}(N_m^{\mathrm{D}}, L_m^{\mathrm{D}})$, where N_m^{D} and L_m^{D} represent the sets of virtual nodes and

virtual links, respectively. The demanded computing capacity of virtual node $j \in N_m^{\mathrm{D}}, m \in M$, is represented by r_{mj}. The set of neighbors of virtual node $j \in N_m^{\mathrm{D}}, m \in M$, is denoted by $B(j)$. The demanded transmission capacity of virtual link $k \in L_m^{\mathrm{D}}, m \in M$, is represented by d_{mk}.

The facility nodes in N^{S} are divided into two types, *primary* facility nodes and *backup* facility nodes, which are given. Let $N_{\mathrm{P}}^{\mathrm{S}}$ and $N_{\mathrm{B}}^{\mathrm{S}}$ represent the sets of primary facility nodes and backup facility nodes, respectively, or $N_{\mathrm{P}}^{\mathrm{S}} \cup N_{\mathrm{B}}^{\mathrm{S}} = N^{\mathrm{S}}$ and $N_{\mathrm{P}}^{\mathrm{S}} \cap N_{\mathrm{B}}^{\mathrm{S}} = \emptyset$. When any service request arrives, all the virtual nodes of corresponding VN are placed in primary facility nodes; all virtual links are mapped to primary substrate paths. The primary computing and transmission resource allocation is given. Let $y_{n^{\mathrm{P}}}^{mj}, j \in N_m^{\mathrm{D}}, m \in M, n^{\mathrm{P}} \in N_{\mathrm{P}}^{\mathrm{S}}$, denote a binary given parameter; $y_{n^{\mathrm{P}}}^{mj} = 1$ means that virtual node j is placed in primary facility node n^{P} and zero otherwise. Backup facility node $n^{\mathrm{B}} \in N_{\mathrm{B}}^{\mathrm{S}}$ is responsible to take the workloads from the protected primary facility nodes when any failure occurs. Note that different virtual nodes from the same VN are not mapped to the same primary facility node for a management purpose [98]. In the same way, this work does not allow the virtual nodes from the same VN to share the backup computing resources in the same backup facility node. Multiple virtual nodes from different VNs can share a backup facility node. Let $p_{n^{\mathrm{P}}}$ denote the failure probability of primary facility node $n^{\mathrm{P}} \in N_{\mathrm{P}}^{\mathrm{S}}$. Usually, backup facility nodes are more infrequently used to handle the workloads than primary facility nodes. This work assumes that the backup facility nodes never fail.

Figure 4.1 shows an example of two VNs embedded in an SN. Each virtual node is placed in a primary facility node. For example, virtual nodes j_3 and j_4 are placed in primary facility node n_3^{P}. Each virtual link is mapped to a primary substrate path. For example, virtual link k_3 is mapped to primary substrate path (l_2, l_3, l_6). For a substrate facility node and a substrate link, the numbers in the parentheses correspond to the maximum computing capacity and the maximum transmission capacity, respectively. For a virtual node and a virtual link, the numbers in the parentheses correspond to the demanded computing capacity and the demanded transmission capacity, respectively.

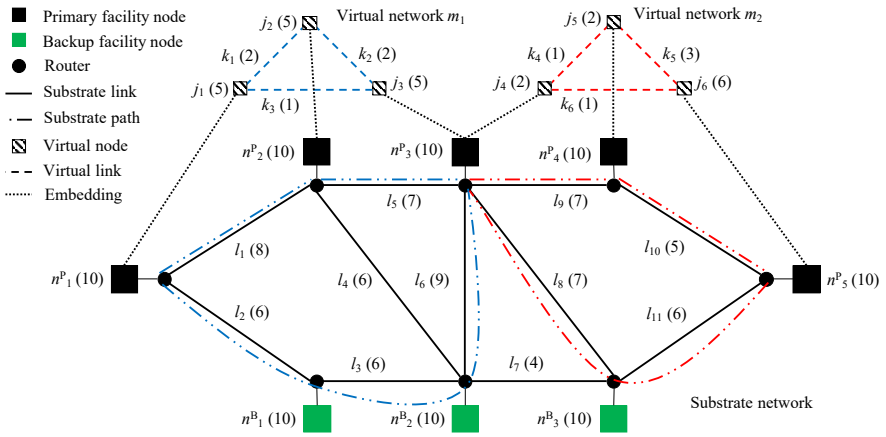


Figure 4.1: Example of VNs in SN.

4.1.2 Backup computing resource allocation for virtual nodes

Let $x^{mj}_{n^{\mathrm{B}}}, j \in N^{\mathrm{D}}_m, m \in M, n^{\mathrm{B}} \in N^{\mathrm{S}}_{\mathrm{B}}$, denote a binary decision variable; $x^{mj}_{n^{\mathrm{B}}}$ is set to one if the backup of r_{mj} is protected by backup facility node n^{B} and zero otherwise. $c^{\mathrm{B}}_{n^{\mathrm{B}}}$ denotes the required backup capacity in backup facility node $n^{\mathrm{B}} \in N^{\mathrm{S}}_{\mathrm{B}}$, which is reserved and remains idle until failures occur. Note that, except for the reserved backup capacity, the remaining capacity in each backup facility node allows us to accept future demands for required backup capacity.

Probabilistic protection

This work considers providing the probabilistic protection, which allows the virtual nodes to share the backup capacity in backup facility nodes, to reduce the required backup computing capacity. This work assumes $p_{n^{\mathrm{P}}} = p, \forall n^{\mathrm{P}} \in N^{\mathrm{S}}_{\mathrm{P}}$, to simplify the discussion. This assumption can be relaxed by using the similar approach in [40]. $X_{n^{\mathrm{P}}}, n^{\mathrm{P}} \in N^{\mathrm{S}}_{\mathrm{P}}$, denotes an independent and identically distributed Bernoulli random variable with parameter p. The survivability

guarantee of probabilistic protection is expressed as,

$$P\left(\sum_{n^{\mathrm{P}}\in N_{\mathrm{P}}^{\mathrm{S}}}\sum_{m\in M}\sum_{j\in N_m^{\mathrm{D}}} X_{n^{\mathrm{P}}} x_{n^{\mathrm{B}}}^{mj} y_{n^{\mathrm{P}}}^{mj} r_{mj} > c_{n^{\mathrm{B}}}^{\mathrm{B}}\right) \leq \epsilon,$$
$$\forall n^{\mathrm{B}} \in N_{\mathrm{B}}^{\mathrm{S}}, \tag{4.1}$$

where ϵ is a given parameter. Equation (4.1) guarantees that the probability that the protection fails due to insufficient backup capacity allocated in a backup facility node is within ϵ.

Figure 4.2 shows the optimal backup computing resource allocation of VNs in Fig. 4.1 when $p = 10^{-2}$ and $\epsilon = 10^{-4}$ without considering any constraint from backup transmission resource allocation. This work takes the backup computing resource allocation in backup facility node n_1^{B}, which protects primary facility nodes n_1^{P} and n_3^{P}, as an example to show how the allocation satisfies (4.1). Table 4.1 shows the probability of each failure case with primary facility nodes n_1^{P} and n_3^{P} and the corresponding required computing capacity on backup facility node n_1^{B} for each failure case. From Table 4.1, we obtain that allocating 5 units of backup computing resource in backup facility node n_1^{B}, as shown in Fig. 4.2, satisfies (4.1), where the probability that the protection fails is $p^2 = 10^{-4} = \epsilon$. Compared to providing the dedicated protection ($\epsilon = 0$), 2 units of backup computing resource are saved in backup facility node n_1^{B} by providing probabilistic protection ($\epsilon = 10^{-4}$).

Table 4.1: Probability and transferred computing capacity for each failure case.

Failure case	Probability	Transferred computing capacity
$X_{n_1^{\mathrm{P}}} = 0, X_{n_3^{\mathrm{P}}} = 0$	$(1-p)^2$	0
$X_{n_1^{\mathrm{P}}} = 0, X_{n_3^{\mathrm{P}}} = 1$	$p(1-p)$	2
$X_{n_1^{\mathrm{P}}} = 1, X_{n_3^{\mathrm{P}}} = 0$	$p(1-p)$	5
$X_{n_1^{\mathrm{P}}} = 1, X_{n_3^{\mathrm{P}}} = 1$	p^2	7

Robust optimization

This work shows how to compute the required backup computing capacity satisfying (4.1). Let $L_{n^{\mathrm{B}}} = \{n^{\mathrm{P}} | \sum_{m\in M}\sum_{j\in N_m^{\mathrm{D}}} x_{n^{\mathrm{B}}}^{mj} y_{n^{\mathrm{P}}}^{mj} \geq 1\}$ represent a set of primary facility nodes protected by backup facility node n^{B}. The number

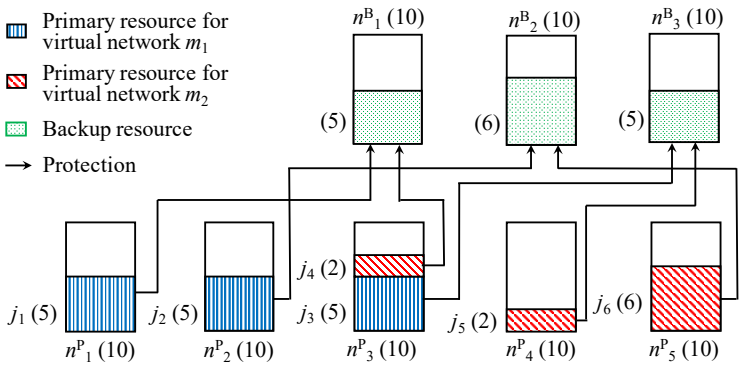


Figure 4.2: Example of backup computing resource allocation with probabilistic protection.

of failed primary facility nodes follows the binomial distribution, where the probability that more than $q \in [0, |L_{n^{\mathrm{B}}}|]$ primary facility nodes in $L_{n^{\mathrm{B}}}$ fail, is expressed as,

$$P\left(\sum_{n^{\mathrm{P}} \in N_{\mathrm{P}}^{\mathrm{S}}} \sum_{m \in M} \sum_{j \in N_m^{\mathrm{D}}} X_{n^{\mathrm{P}}} x_{n^{\mathrm{B}}}^{mj} y_{n^{\mathrm{P}}}^{mj} > q\right)$$
$$= \sum_{\sigma=q+1}^{|L_{n^{\mathrm{B}}}|} \binom{|L_{n^{\mathrm{B}}}|}{\sigma} p^{\sigma} (1-p)^{|L_{n^{\mathrm{B}}}|-\sigma}. \tag{4.2}$$

For a backup facility node, it is uncertain which protected primary facility nodes in $L_{n^{\mathrm{B}}}$ fail, which leads to uncertainty in the required backup computing capacity. Similar to the works in [37, 40], this work adopts the idea of robust optimization to consider resisting uncertainty. Let $\Gamma_{n^{\mathrm{B}}}$ denote the minimum value of q, where the value of (4.2) is not greater than ϵ. Let $S_{n^{\mathrm{B}}}$ be a set of $\Gamma_{n^{\mathrm{B}}}$ primary facility nodes in $L_{n^{\mathrm{B}}}$ with the largest computing capacities protected by backup facility node n^{B}. The required backup computing capacity is computed by,

$$c_{n^{\mathrm{B}}}^{\mathrm{B}} = \sum_{n^{\mathrm{P}} \in S_{n^{\mathrm{B}}}} \sum_{m \in M} \sum_{j \in N_m^{\mathrm{D}}} x_{n^{\mathrm{B}}}^{mj} y_{n^{\mathrm{P}}}^{mj} r_{mj}, \tag{4.3}$$

which satisfies the probabilistic protection guarantee in (4.1).

4.1.3 Backup transmission resource allocation for virtual links

When some primary facility nodes fail, the embedded virtual nodes are moved to the corresponding backup facility nodes. As a result, the virtual links connected to the moved virtual nodes need to be remapped to some backup substrate paths. When the backup facility node for each virtual node is determined, the routes of backup paths for virtual links need to be determined and a certain backup transmission capacity needs to be reserved at the same time.

This work analyzes the types of backup substrate paths required to be prepared for each virtual link. Any virtual node has two states; in one state, the corresponding primary facility node fails, and, in the other state, the primary facility node does not fail. Accordingly, there are four cases for the states of a pair of connected virtual nodes, which leads to one primary substrate path and three types of backup substrate paths required to be prepared for the virtual link. In case 1, the corresponding two primary facility nodes do not fail, and the virtual link is mapped in its primary path, which is given. In cases 2 and 3, one of the primary facility node fails and the other does not. Two backup paths, which are defined as the first and second backup paths, are required to be determined for cases 2 and 3, respectively. In case 4, the two primary facility nodes fail simultaneously, and the third backup path is required. As a result, the total number of backup substrate paths required to be prepared for all the virtual links is $3\sum_{m\in M}|L_m^{\mathrm{D}}|$, where $\sum_{m\in M}|L_m^{\mathrm{D}}|$ is the total number of virtual links.

Let $\alpha_{nn'}^{mjj'}, m \in M, j \in N_m^{\mathrm{D}}, j' \in B(j), n \in N^{\mathrm{S}}, n' \in A(n)$, denote a binary decision variable; $\alpha_{nn'}^{mjj'}$ is set to one if the first or second backup path of virtual link $k = (j, j') \in L_m^{\mathrm{D}}$ is routed through substrate link $l = (n, n') \in L^{\mathrm{S}}$ and zero otherwise, where the corresponding primary facility node of virtual node j fails and that of virtual node j' does not fail. Here we do not need to specify variables $\alpha_{nn'}^{mjj'}$ for the first or second backup path of virtual link $k = (j, j')$. For example, for virtual link $k_1 = (j_1, j_2)$ in Fig. 4.1, one of $\alpha_{nn'}^{m_1 j_1 j_2}$ and $\alpha_{nn'}^{m_1 j_2 j_1}$ corresponds to the first backup path, and the other corresponds to the second backup path. Let $\beta_{nn'}^{mk}, m \in M, k \in L_m^{\mathrm{D}}, n \in N^{\mathrm{S}}, n' \in A(n)$, denote a binary decision variable; $\beta_{nn'}^{mk}$ is set to one if the third backup path of virtual

link $k = (j, j') \in L_m^{\mathrm{D}}$ is routed through substrate link $l = (n, n') \in L^{\mathrm{S}}$ and zero otherwise, where the corresponding primary facility nodes of virtual nodes j and j' fail simultaneously.

Note that (n, n') and (n', n) represent the same substrate link, which is undirected in this thesis by default. This work considers it as directed only for $\alpha_{nn'}^{mjj'}$ and $\beta_{nn'}^{mk}$, which means that $\alpha_{nn'}^{mjj'}$ ($\beta_{nn'}^{mk}$) and $\alpha_{n'n}^{mjj'}$ ($\beta_{n'n}^{mk}$) are two different variables. This work gives the following two constraints on $\alpha_{nn'}^{mjj'}$ and $\beta_{nn'}^{mk}, m \in M, j \in N_m^{\mathrm{D}}, j' \in B(j), k \in L_m^{\mathrm{D}}, n \in N^{\mathrm{S}}, n' \in A(n)$, respectively, to avoid any repeated routing, where a backup path is routed through the same substrate link twice.

$$\alpha_{nn'}^{mjj'} + \alpha_{n'n}^{mjj'} \leq 1, \forall m \in M, j \in N_m^{\mathrm{D}}, j' \in B(j), l \in L^{\mathrm{S}}. \tag{4.4}$$

$$\beta_{nn'}^{mk} + \beta_{n'n}^{mk} \leq 1, \forall m \in M, k \in L_m^{\mathrm{D}}, l \in L^{\mathrm{S}}. \tag{4.5}$$

Flow constraints

A substrate path may include several primary facility nodes in $N_{\mathrm{P}}^{\mathrm{S}}$ and backup facility nodes in $N_{\mathrm{B}}^{\mathrm{S}}$. For the first (second) backup path of a virtual link, this work considers the backup facility node of virtual node whose primary facility node fails as the source node in the backup path; the primary facility node, which does not fail, of the other virtual node is considered as the destination node in the backup path. Clearly, both source and destination nodes of the third backup path of a virtual link are backup facility nodes.

The flow constraints for the first and second backup paths are expressed by,

$$\sum_{n' \in A(n^{\mathrm{B}})} \alpha_{n^{\mathrm{B}}n'}^{mjj'} - \sum_{n' \in A(n^{\mathrm{B}})} \alpha_{n'n^{\mathrm{B}}}^{mjj'} = x_{n^{\mathrm{B}}}^{mj},$$
$$\forall m \in M, j \in N_m^{\mathrm{D}}, j' \in B(j), n^{\mathrm{B}} \in N_{\mathrm{B}}^{\mathrm{S}}, \tag{4.6}$$

$$\sum_{n' \in A(n^{\mathrm{P}})} \alpha_{n^{\mathrm{P}}n'}^{mjj'} - \sum_{n' \in A(n^{\mathrm{P}})} \alpha_{n'n^{\mathrm{P}}}^{mjj'} = -y_{n^{\mathrm{P}}}^{mj'},$$
$$\forall m \in M, j \in N_m^{\mathrm{D}}, j' \in B(j), n^{\mathrm{P}} \in N_{\mathrm{P}}^{\mathrm{S}}. \tag{4.7}$$

Similarly, the flow constraints for the third backup paths are expressed by,

$$\sum_{n' \in A(n^{\mathrm{B}})} \beta^{mk}_{n^{\mathrm{B}} n'} - \sum_{n' \in A(n^{\mathrm{B}})} \beta^{mk}_{n' n^{\mathrm{B}}} = x^{mj}_{n^{\mathrm{B}}} - x^{mj'}_{n^{\mathrm{B}}},$$
$$\forall m \in M, k \in L^{\mathrm{D}}_m, n^{\mathrm{B}} \in N^{\mathrm{S}}_{\mathrm{B}}, \tag{4.8}$$

$$\sum_{n' \in A(n^{\mathrm{P}})} \beta^{mk}_{n^{\mathrm{P}} n'} - \sum_{n' \in A(n^{\mathrm{P}})} \beta^{mk}_{n' n^{\mathrm{P}}} = 0, \forall m \in M, k \in L^{\mathrm{D}}_m, n^{\mathrm{P}} \in N^{\mathrm{S}}_{\mathrm{P}}. \tag{4.9}$$

Capacity constraints

Let $u^{mk}_l, m \in M, k \in L^{\mathrm{D}}_m, l \in L^{\mathrm{S}}$, denote a binary given parameter; $u^{mk}_l = 1$ means that the primary substrate path for virtual link k is routed through substrate link l and zero otherwise. Let b^{P}_l represent the transmission capacity in substrate link $l \in L^{\mathrm{S}}$ used for primary substrate paths, which can be expressed by,

$$b^{\mathrm{P}}_l = \sum_{m \in M} \sum_{k \in L^{\mathrm{D}}_m} u^{mk}_l d_{mk}, \forall l \in L^{\mathrm{S}}. \tag{4.10}$$

Let b^{B}_l represent the required backup capacity in substrate link $l \in L^{\mathrm{S}}$. The transmission capacity constraint for substrate link $l \in L^{\mathrm{S}}$ is expressed by,

$$b^{\mathrm{P}}_l + b^{\mathrm{B}}_l \leq b_l, \forall l \in L^{\mathrm{S}}. \tag{4.11}$$

4.1.4 Problem formulation

This work formulates the backup resource allocation for VNs (BRAVN) problem as the following optimization problem to minimize the total required backup computing capacity.

$$\min \quad \sum_{n^{\mathrm{B}} \in N^{\mathrm{S}}_{\mathrm{B}}} c^{\mathrm{B}}_{n^{\mathrm{B}}} \tag{4.12a}$$

$$s.t. \quad \text{Eqs. (4.1), (4.4) – (4.11)} \tag{4.12b}$$

$$\sum_{n^{\mathrm{B}} \in N^{\mathrm{S}}_{\mathrm{B}}} x^{mj}_{n^{\mathrm{B}}} = 1, \forall m \in M, j \in N^{\mathrm{D}}_m \tag{4.12c}$$

$$\sum_{j \in N^{\mathrm{D}}_m} x^{mj}_{n^{\mathrm{B}}} \leq 1, \forall m \in M, n^{\mathrm{B}} \in N^{\mathrm{S}}_{\mathrm{B}} \tag{4.12d}$$

$$c^{\mathrm{B}}_{n^{\mathrm{B}}} \leq c_{n^{\mathrm{B}}}, \forall n^{\mathrm{B}} \in N_{\mathrm{B}} \tag{4.12e}$$

$$\alpha^{mjj'}_{nn'} \in \{0,1\}, \forall m \in M,$$
$$j \in N^{\mathrm{D}}_m, j' \in B(j), n \in N^{\mathrm{S}}, n' \in A(n) \tag{4.12f}$$

$$\beta^{mk}_{nn'} \in \{0,1\},$$
$$\forall m \in M, k \in L^{\mathrm{D}}_m, n \in N^{\mathrm{S}}, n' \in A(n) \tag{4.12g}$$

$$x^{mj}_{n^{\mathrm{B}}} \in \{0,1\}, \forall m \in M, j \in N^{\mathrm{D}}_m, n^{\mathrm{B}} \in N_{\mathrm{B}}. \tag{4.12h}$$

Equation (4.12a) minimizes the total required backup computing capacity. Equation (4.12c) ensures that each virtual node is protected by a backup facility node. Equation (4.12d) indicates that a backup facility node protects at most one virtual node in the same VN. In other words, more than one virtual node from the same VN is protected by different backup facility nodes. Equation (4.12e) is the computing capacity constraint, which indicates that $c^{\mathrm{B}}_{n^{\mathrm{B}}}$ does not exceed $c_{n^{\mathrm{B}}}$. Equations (4.12f), (4.12g), and (4.12h) show the ranges of decision variables.

Adopting the techniques presented in Chapter 3, this work transforms the above optimization problem to an MILP problem without considering backup transmission resource sharing. Clearly, if two backup paths are not used simultaneously, they can share a certain backup transmission capacity, which reduces the required backup transmission capacity. Less required backup transmission capacity can lead to more feasible solutions for backup computing resource allocation. As a result, the required backup computing capacity can be reduced. This work analyzes two types of backup transmission resource sharing to compute the minimum required backup transmission capacity in Section 4.2.

4.2 Analyses for backup transmission resource sharing

In this section, this work firstly explains two kinds of backup transmission resource sharing presented in [99,100] with considering single substrate facility node failure. Then this work generally analyzes the two types of sharing in multiple VNs cases with considering multiple substrate facility node failures to minimize the required backup transmission capacity. This work discusses three

specific questions generated from the analyses. Finally, this work classifies the proposed model in terms of the degree of backup transmission resource sharing. Table 4.2 summarizes the frequently used notations.

Table 4.2: List of frequently used notations in Chapter 4.

Notation	Meaning
V	Set of all primary and backup paths
$V^{\mathrm{P}} \subseteq V$	Set of all primary paths
$V^{\mathrm{B}} \subseteq V$	Set of all backup paths
$W^{\mathrm{P}} \subseteq V^{\mathrm{P}}$	Subset of primary paths
$W^{\mathrm{B}} \subseteq V^{\mathrm{B}}$	Subset of backup paths
v^{P}	Primary path
v^{B}	Backup path
$V_l \subseteq V$	Set of all substrate paths that include substrate link l
$\mathbb{W}_l^{\mathrm{B}}$	Set of all maximal unexclusive subsets on substrate link l

4.2.1 Cross-sharing and backup-sharing

The works in [99, 100] studied protection strategies against single substrate facility node failure. This work takes VN m_1 in Fig. 4.1 as an example, where the primary paths of virtual links k_1, k_2, and k_3, are (l_1), (l_5), and (l_2, l_3, l_6), respectively. The computing resources for VN m_1 in n_1^{P}, n_2^{P}, and n_3^{P} are protected by n_1^{B}, n_2^{B}, and n_3^{B}, respectively, as shown in Fig. 4.2. For a single failure in facility node n_1^{P}, let substrate paths (l_2, l_1) and (l_3, l_6) be the backup substrate paths for virtual links k_1 and k_3, respectively. For a single failure in facility node n_2^{P}, let substrate paths (l_2, l_3) and (l_6) be the backup substrate paths for virtual links k_1 and k_2, respectively. For a single failure in facility node n_3^{P}, let substrate paths (l_7, l_4) and (l_7, l_4, l_1) be the backup substrate paths for virtual links k_2 and k_3, respectively. Figure 4.3 shows all the backup and primary paths of VN m_1 against single facility node failure, where some network information in Fig. 4.1 is omitted.

This work discusses the backup transmission capacity of VN m_1 required in substrate link l_2 against any single failure. Only the two backup paths of virtual link k_1, which demands 2 units of transmission capacity in a substrate path, are routed through substrate link l_2. If we do not consider any backup transmission capacity sharing, in total 4 units of transmission capacity are required to be reserved for the two backup paths. Since the transmission capacity used for primary substrate path (l_2, l_3, l_6) of virtual link k_3 is released

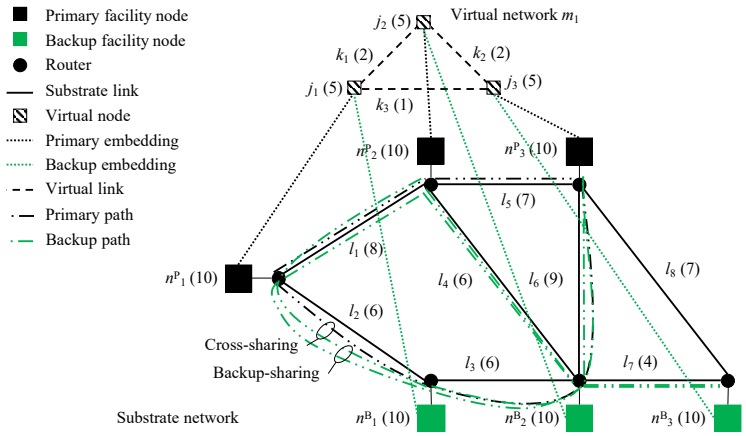


Figure 4.3: Backup and primary paths of VN m_1.

when primary facility node n_1^{P} fails, it can be used for backup substrate path (l_2, l_1) of virtual link k_1. As a result, 1 unit of backup transmission resource is saved, and only 1 unit of backup transmission resource is required to be reserved in substrate link l_2 for virtual link k_1 against the single failure in primary facility node n_1^{P}. This kind of sharing is called *cross-sharing* between primary and backup substrate paths, as shown in Fig. 4.3. To recover virtual link k_1 in the case of single failure in primary facility node n_2^{P}, since 1 unit of backup transmission resource is already reserved in substrate link l_2 against the single failure in primary facility node n_1^{P}, only 1 unit of backup transmission resource is additionally needed, which means that another 1 unit of backup transmission resource is saved. Such type of sharing is known as *backup-sharing* between backup paths which correspond to different single failure cases, as shown in Fig. 4.3. Therefore, in total $4 - 2 = 2$ units of backup transmission resource are required in substrate link l_2 for VN m_1 against any single failure in primary facility nodes.

4.2.2 Backup transmission resource sharing in multiple virtual networks with multiple substrate facility node failures

In this problem, in total four substrate paths, which consists of one primary path and three backup paths, need to be prepared for each virtual link. Let V

denote the set of substrate paths required to be prepared, including primary and backup paths, for all virtual links, and the size of V is $|V| = 4\sum_{m\in M}|L_m^{\mathrm{D}}|$. Let $V^{\mathrm{B}} \subseteq V$ and $V^{\mathrm{P}} \subseteq V$, where $V = V^{\mathrm{B}} \cup V^{\mathrm{P}}$, denote sets of all backup and primary paths, respectively.

To describe the relationship between any two prepared substrate paths in V, this work gives the following definition:

Definition 1 *If prepared substrate paths $v \in V$ and $v' \in V$ are not used simultaneously, substrate paths v and v' are mutually exclusive. Otherwise, they are not mutually exclusive.*

Obviously, for a primary path and a backup path which are mutually exclusive, the transmission capacity of primary path can be used for the backup path after the releasing of primary path; two backup paths which are mutually exclusive can share the same reserved backup transmission capacity. For example, since any two of the four prepared substrate paths of a virtual link are not used simultaneously at any time, they are mutually exclusive.

Backup transmission resource sharing, such as cross-sharing and backup-sharing, cannot be performed between a backup path and another path, if they are not mutually exclusive. For a subset of backup paths, $W^{\mathrm{B}} \subseteq V^{\mathrm{B}}$, this work gives the following two definitions:

Definition 2 *W^B is an unexclusive subset of backup paths of V^B, if any two backup paths in W^B are not mutually exclusive or there is only one backup path in W^B.*

Definition 3 *W^B is a maximal unexclusive subset of backup paths of V^B, if W^B is an unexclusive subset of backup paths of V^B and there is no backup path $v^B \in V^B \backslash W^B$ such that $W^B \cup \{v^B\}$ is still an unexclusive subset of backup paths of V^B.*

Let $V_l \subseteq V$ represent the set of all the prepared substrate paths which include substrate link $l \in L^{\mathrm{S}}$. Let $V_l^{\mathrm{B}} \subseteq V_l$ and $V_l^{\mathrm{P}} \subseteq V_l$, where $V_l = V_l^{\mathrm{B}} \cup V_l^{\mathrm{P}}$, $l \in L^{\mathrm{S}}$, denote sets of all the backup and primary paths in V_l, respectively. Let $W_l^{\mathrm{B}} \subseteq V_l^{\mathrm{B}}$, $l \in L^{\mathrm{S}}$, be an unexclusive subset of backup paths of V_l^{B}. There is no backup-sharing between any two backup paths in W_l^{B}, $l \in L^{\mathrm{S}}$.

This work analyzes the situations of cross-sharing between the primary paths in V_l^{P}, $l \in L^{\mathrm{S}}$, and the backup paths in W_l^{B} based on the framework of bipartite graph in graph theory to compute the total required backup transmission capacity of W_l^{B} in substrate link l. Here, backup path $v^{\mathrm{B}} \in W_l^{\mathrm{B}}$, $l \in L^{\mathrm{S}}$, may mutually exclusive with multiple primary paths in V_l^{P}. Contrarily, primary path $v^{\mathrm{P}} \in V_l^{\mathrm{P}}$, $l \in L^{\mathrm{S}}$, may mutually exclusive with multiple backup paths in W_l^{B}. Let $W_l^{\mathrm{P}} \subseteq V_l^{\mathrm{P}}$, $l \in L^{\mathrm{S}}$, represent the set of all the primary paths in V_l^{P}, each of which is mutually exclusive with at least one backup path in W_l^{B}.

The relationships between backup paths in W_l^{B}, $l \in L^{\mathrm{S}}$, and primary paths in W_l^{P} can be represented by a bipartite graph, where the backup and primary paths represent the nodes in two sides, respectively, and an edge exists between a backup path and a primary path if they are mutually exclusive. Figure 4.4 shows an example of three backup paths in W_l^{B}, $l \in L^{\mathrm{S}}$, and three primary paths in W_l^{P}.

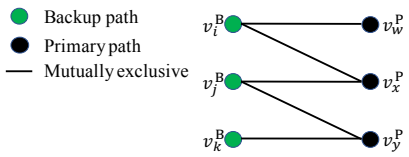


Figure 4.4: Example of relationships between backup and primary paths represented by bipartite graph.

Given a bipartite graph, this work considers the situations of cross-sharing between backup and primary paths in the bipartite graph to judge if the total released transmission capacity from primary paths is sufficient for the total demanded transmission capacity of backup paths. If it is not, extra transmission capacity is required for backup paths in addition to the total released transmission capacity from primary paths so that all the backup paths can protect all the primary paths. This work computes the required backup transmission capacity. Note that a backup path and a primary path which are in the same bipartite graph and are mutually exclusive may not correspond to the same virtual link.

For example, for the bipartite graph in Fig. 4.4, let the demanded transmission capacities of virtual links corresponding to backup paths v_i^{B}, v_j^{B}, and

v_k^{B} be 1, 2, and 3 units of transmission capacity, respectively; let all the released transmission capacities of primary paths v_w^{P}, v_x^{P}, and v_y^{P} be 1 unit of transmission capacity. Consider multiple facility failures which activate all the backup paths and simultaneously release all the primary paths in the bipartite graph, where the demanded transmission capacity of a node can use the released transmission capacities in other nodes which connect with the node. The 1 unit of demanded transmission capacity of backup path v_i^{B} can use the 1 unit of transmission capacity of primary path v_w^{P} after its releasing. Similarly, the two 1 unit of released transmission capacity from primary path v_x^{P} and v_y^{P} can be used for the 2 units of demanded transmission capacity of backup path v_j^{B}. As a result, there is no more released transmission capacity from primary path v_y^{P} to be used for the demanded transmission capacity of backup path v_k^{B}. Therefore, 3 units of transmission capacity, after considering cross-sharing, are additionally required to be reserved for all the backup paths.

The total required backup transmission capacity of W_l^{B} in substrate link $l \in L^{\mathrm{S}}$, $b_{W_l^{\mathrm{B}}}^{\mathrm{B}}$, is equivalent to the total transmission capacity additionally required for all the backup paths in the corresponding bipartite graph. Let $\mathbb{W}_l^{\mathrm{B}}$, $l \in L^{\mathrm{S}}$, represent the set of all the maximal unexclusive subsets of backup paths of V_l^{B}. Note that the same backup path may exist in some different maximal unexclusive subsets in $\mathbb{W}_l^{\mathrm{B}}$, $l \in L^{\mathrm{S}}$. Therefore, the required backup capacity in substrate link $l \in L^{\mathrm{S}}$, b_l^{B}, equals the required backup capacity of maximal unexclusive subset which is with the greatest required backup capacity among all the maximal unexclusive subsets in $\mathbb{W}_l^{\mathrm{B}}$, or it is expressed by,

$$b_l^{\mathrm{B}} = \max_{W_l^{\mathrm{B}} \in \mathbb{W}_l^{\mathrm{B}}} b_{W_l^{\mathrm{B}}}^{\mathrm{B}}. \tag{4.13}$$

Actually, there are three remaining questions for computing b_l^{B}. Question 1 is what is the specific conditions to judge whether two prepared substrate paths in V are mutually exclusive. Based on the judging conditions, question 2 is how to obtain all the maximal unexclusive subsets of a given set of backup paths. In a substrate link, there is a corresponding set of primary paths for each maximal unexclusive subset of backup paths. This work derives a bipartite graph to represent the relationships between the primary and backup paths in the two sets. Question 3 is how to compute the total required backup

transmission capacity of backup paths in a given bipartite graph. This work analyzes the three questions in Sections 4.2.3, 4.2.3, and 4.2.3, respectively.

4.2.3 Analyses for three questions

Analyses for question 1

This work shows the specific conditions to judge whether two prepared substrate paths in V are mutually exclusive. According to Definition 1, they are equivalent to the conditions to judge whether two prepared substrate paths in V are used simultaneously.

Given two virtual nodes connected by a virtual link and their states, a corresponding prepared substrate path is specified, and vice versa. Two prepared substrate paths may correspond to the same pair of virtual nodes but with different states. Let $k^v = (f^v, j^v) \in L_m^{\mathrm{D}}, f^v \in N_m^{\mathrm{D}}, j^v \in N_m^{\mathrm{D}}, m \in M$, represent the corresponding virtual link of prepared substrate path $v \in V$, where f^v and j^v are two corresponding virtual nodes. Let $g_{j^v} \in N_{\mathrm{P}}^{\mathrm{S}}$ represent the primary facility node where virtual node $j^v \in N_m^{\mathrm{D}}, m \in M, v \in V$, is initially embedded, which is given, or $y_{g_{j^v}}^{mj^v} = 1$. Consider s_{j^v} as a binary index to represent the state of virtual node $j^v \in N_m^{\mathrm{D}}, m \in M, v \in V$; $s_{j^v} = 1$ expresses that virtual node j^v is in the state that the corresponding primary facility node g_{j^v} fails and zero otherwise.

For prepared substrate paths $v \in V$ and $v' \in V$, consider a situation that there is at least one pair of virtual nodes which are initially embedded in the same primary facility node and are with different states, or $\exists\ j^v$ and $j^{v'}$ such that $g_{j^v} = g_{j^{v'}}$ and $s_{j^v} \neq s_{j^{v'}}$. Note that the pair of virtual nodes may be the same virtual node with different states or two different virtual nodes of different VNs. In this situation, one prepared substrate path is active only when primary facility node g_{j^v} does not fail, and the other path is active only when primary facility node g_{j^v} fails. It means that prepared substrate paths $v \in V$ and $v' \in V$ are not used simultaneously. Hence, they are mutually exclusive. On the contrary, if there is no such a pair of virtual nodes, the two prepared substrate paths can be used simultaneously, and they are not mutually exclusive.

Analyses for question 2

Given the set of all the backup paths including substrate link $l \in L^{\mathrm{S}}$, V_l^{B}, this work analyzes how to obtain all maximal unexclusive subsets of backup paths of V_l^{B}. Consider V_l^{B} as an undirected graph, where each backup path in V_l^{B} represent a node and there is an edge between two nodes if the two corresponding backup paths are *not* mutually exclusive. Let $G^{V_l^{\mathrm{B}}}$ represent the graph corresponding to V_l^{B}. Note that the condition of existing an edge here is opposite to that in the bipartite graphs defined in Sections 4.2.2 and 4.2.3. Figure 4.5(a) shows an example of V_l^{B} consisting of seven backup paths, where an edge connecting with two backup paths with a dotted line denotes that the two backup paths are not mutually exclusive.

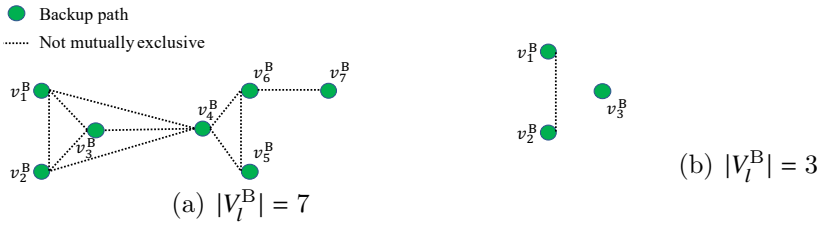


Figure 4.5: Examples of V_l^{B}.

In graph theory, a *clique* in an undirected graph is a subset of nodes, where each two nodes are adjacent [101]. A maximal clique is a clique that cannot be extended by adding any node which is adjacent with all the existing nodes in the clique. There are several cliques in Fig. 4.5(a), such as $\{v_1^{\mathrm{B}}, v_2^{\mathrm{B}}, v_3^{\mathrm{B}}\}$. There are three maximal cliques in Fig. 4.5(a), which are $\{v_1^{\mathrm{B}}, v_2^{\mathrm{B}}, v_3^{\mathrm{B}}, v_4^{\mathrm{B}}\}$, $\{v_4^{\mathrm{B}}, v_5^{\mathrm{B}}, v_6^{\mathrm{B}}\}$, and $\{v_6^{\mathrm{B}}, v_7^{\mathrm{B}}\}$.

Clearly, the concepts of unexclusive subset and maximal unexclusive subset of backup paths in this thesis are similar to the concepts of clique and maximal clique in graph theory, respectively. The only difference is that an unexclusive or maximal unexclusive subset can contain only one backup path, but a clique or maximal clique contains at least two nodes. For example, in Fig. 4.5, any maximal unexclusive subsets of V_l^{B} containing at least two backup paths constructs a maximal clique in $G^{V_l^{\mathrm{B}}}$; in Fig. 4.5(b), $\{v_3^{\mathrm{B}}\}$ is a maximal unexclusive subset of V_l^{B} but not a maximal clique of $G^{V_l^{\mathrm{B}}}$. Let $\delta(V_l^{\mathrm{B}})$ and

$\delta'(G^{V_l^{\mathrm{B}}})$ represent the number of maximal unexclusive subsets of V_l^{B} and the number of maximal cliques of $G^{V_l^{\mathrm{B}}}$, respectively.

Let $\Lambda(t)$ denote the set containing t backup paths, where the number of maximal unexclusive subsets of $\Lambda(t)$ is maximized among all sets with t backup paths, or $\Lambda(t) = \mathrm{argmax}\{\delta(V_l^{\mathrm{B}})|V_l^{\mathrm{B}} \subseteq V^{\mathrm{B}} : |V_l^{\mathrm{B}}| = t\}$. Based on the studies for maximal clique, this work provides the flowing analyses for maximal unexclusive set, which can be used to design algorithms finding all maximal unexclusive subsets of backup paths of V_l^{B}.

Lemma 1 *In $\Lambda(t)$ with $t \geq 2$, there is at least one pair of backup paths which are mutually exclusive, or there is at least one pair of nodes which are nonadjacent in $G^{\Lambda(t)}$.*

Proof : This work uses contradiction to prove Lemma 1.

Suppose that any two backup paths are not mutually exclusive in $\Lambda(t)$. As a result, $\Lambda(t)$ itself is a maximal unexclusive subset of backup paths of $\Lambda(t)$, which indicates $\delta(\Lambda(t)) = 1$. However, for V_l^{B} with $|V_l^{\mathrm{B}}| = t$, where any two backup paths are mutually exclusive, we have $\delta(V_l^{\mathrm{B}}) = t > \delta(\Lambda(t))$. It contradicts that $\Lambda(t) = \mathrm{argmax}\{\delta(V_l^{\mathrm{B}})|V_l^{\mathrm{B}} \subseteq V^{\mathrm{B}} : |V_l^{\mathrm{B}}| = t\}$. Therefore, there is at least one pair of backup paths which are mutually exclusive in $\Lambda(t)$ when $t \geq 2$. □

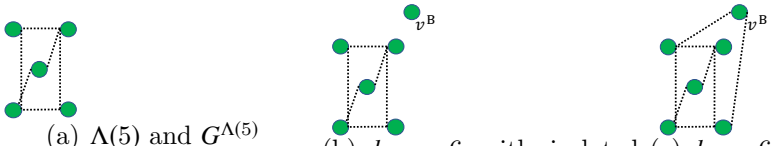
Theorem 2 *When $t \geq 5$, $G^{\Lambda(t)}$ is a connected graph.*

Proof : This work uses induction to prove Theorem 2.

When $t = 5$, we obtain that $G^{\Lambda(5)}$ is with the topology shown in Fig. 4.6(a), where $G^{\Lambda(5)}$ is a connected graph.

Suppose that Theorem 2 holds for $t = k - 1$. When $t = k$, this work uses contradiction to prove that $G^{\Lambda(k)}$ is a connected graph. Suppose that $\Lambda(k)$ contains backup path $v^{\mathrm{B}} \in V_l^{\mathrm{B}}$ which is mutually exclusive with any other backup path in $\Lambda(k)$, or $G^{\Lambda(k)}$ contains an isolated node which is nonadjacent with any other node in $G^{\Lambda(k)}$. We obtain $\delta(\Lambda(k)) = 1 + \delta(\Lambda(k) \setminus \{v^{\mathrm{B}}\})$. Since $\delta(\Lambda(k)) = \max_{V_l^{\mathrm{B}} \subseteq V^{\mathrm{B}}:|V_l^{\mathrm{B}}|=k} \delta(V_l^{\mathrm{B}})$, we have $\Lambda(k) \setminus \{v^{\mathrm{B}}\} = \Lambda(k-1)$. It indicates that $G^{\Lambda(k)}$ consists of the isolated node and the connected graph of $G^{\Lambda(k-1)}$, such as Fig 4.6(b) for $k = 6$. Based on Lemma 1, there is at least one pair

of nodes which are nonadjacent in $G^{\Lambda(k-1)}$. Connecting the isolated node with the two nodes which are nonadjacent leads to V_l^{B}, such as Fig 4.6(c) for $k = 6$, where $|V_l^{\mathrm{B}}| = k$ and $\delta(V_l^{\mathrm{B}}) = 2 + \delta(\Lambda(k-1)) > \delta(\Lambda(k))$. It contradicts that $\Lambda(k) = \mathrm{argmax}\{\delta(V_l^{\mathrm{B}}) | V_l^{\mathrm{B}} \subseteq V^{\mathrm{B}} : |V_l^{\mathrm{B}}| = k\}$. Hence, $G^{\Lambda(k)}$ is a connected graph.



(a) $\Lambda(5)$ and $G^{\Lambda(5)}$ (b) $k = 6$ with isolated node (c) $k = 6$ with connected graph

Figure 4.6: Examples of V_l^{B} for Theorem 2; same symbols with Fig. 4.5 are used.

Therefore, $G^{\Lambda(t)}$ is a connected graph when $t \geq 5$. □

Theorem 3 *Given V_l^B with $|V_l^B| = t$, there are at most $3^{\frac{t}{3}}$ maximal unexclusive subsets of backup paths of V_l^B.*

Proof : When $t = 1, 2, 3,$ and 4, we obtain $\delta(\Lambda(t)) = 1, 2, 3,$ and 4, respectively, each of which supports Theorem 3.

When $t \geq 5$, Theorem 2 indicates that $\delta(\Lambda(t)) = \delta'(G^{\Lambda(t)})$. The work in [102] proved that there are at most $3^{\frac{n}{3}}$ maximal cliques in a graph with n nodes, which indicates that $\delta'(G^{\Lambda(t)}) \leq 3^{\frac{t}{3}}$ and $\delta(\Lambda(t)) \leq 3^{\frac{t}{3}}$.

Therefore, given V_l^{B} containing t backup paths, there are at most $3^{\frac{t}{3}}$ maximal unexclusive subsets of V_l^{B}. □

Analyses for question 3

This work computes the total required backup transmission capacity of backup paths in a given bipartite graph by solving a minimum cost flow (MCF) problem [103]. Consider a given bipartite graph, such as the one in Fig. 4.4, where backup path set W_l^{B} and primary path set W_l^{P} represent the sets of nodes in two sides, respectively. d_v denotes the demanded transmission capacity of virtual link corresponding to substrate path $v \in W_l^{\mathrm{B}} \cup W_l^{\mathrm{P}}$. Consequently, $d_{v^{\mathrm{B}}}$ and

$d_{v^{\mathrm{P}}}$ represent the demanded transmission capacity of backup path $v^{\mathrm{B}} \in W_l^{\mathrm{B}}$ and the released transmission capacity of primary path $v^{\mathrm{P}} \in W_l^{\mathrm{P}}$, respectively.

Based on the given bipartite graph, this work constructs a directed graph by adding sets of nodes and edges and by specifying the information of each edge. This work adds three nodes, which are a source node, a middle node, and a destination node. This work adds directed edges from the source node to backup path $v^{\mathrm{B}} \in W_l^{\mathrm{B}}$, from backup path v^{B} to the middle node, from primary path $v^{\mathrm{P}} \in W_l^{\mathrm{P}}$ to the destination node, and from the middle node to the destination node, with the edge costs and capacities of $(0, d_{v^{\mathrm{B}}})$, $(0, d_{v^{\mathrm{B}}})$, $(0, d_{v^{\mathrm{P}}})$, and $(1, \sum_{v^{\mathrm{B}} \in W_l^{\mathrm{B}}} d_{v^{\mathrm{B}}})$, respectively. The edge costs and capacities for an edge from backup path v^{B} to primary path v^{P} are considered as $(0, d_{v^{\mathrm{B}}})$. Figure 4.7 shows the constructed directed graph based on the bipartite graph in Fig. 4.4.

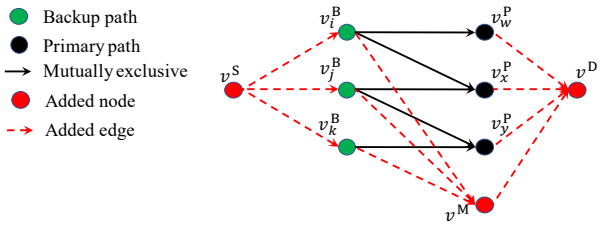


Figure 4.7: Constructed directed graph based on given bipartite graph in Fig. 4.4.

For the MCF problem based on the constructed directed graph with considering the flow requesting the traffic volume of $\sum_{v^{\mathrm{B}} \in W_l^{\mathrm{B}}} d_{v^{\mathrm{B}}}$ from the source node to the destination node, the optimal objective value is the flow cost on the edge from the middle node to the destination node, which is equivalent to the total required backup transmission capacity of backup paths in the given bipartite graph. In literature, several polynomial time algorithms, such as the one presented in [103], can be adopted to solve the MCF problem.

4.2.4 Proposed model with different degrees of backup transmission resource sharing

Different degrees of backup transmission resource sharing can be considered in the proposed backup resource allocation model for VNs. The proposed model with full backup transmission resource sharing (P-FTS) minimizes the required backup transmission capacity. In practical applications, computing the exact minimized required backup transmission capacity may not be necessary as long as a robust one satisfies the transmission capacity constraint. The proposed model with limited backup transmission resource sharing (P-LTS) considers the cross-sharing and backup-sharing among paths corresponding to the same virtual link. The proposed model with non backup transmission resource sharing (P-NTS) requires the simplest computation with reserving the most backup transmission capacity.

4.3 Heuristic algorithm

This work focuses on the heuristic algorithm to solve the BRAVN problem with considering full backup transmission resource sharing, or P-FTS. For P-LTS and P-NTS, the introduced algorithm can be easily modified to fit each case.

4.3.1 Framework

This work presents a disjoint backup allocating (DBA) algorithm (see Algorithm 1). In the DBA algorithm, this work first obtains a set of backup computing resource allocations, which is denoted by $\boldsymbol{F}$, without considering the capacity constraint of (4.12e) and constraints from backup transmission resource allocation through a multiple-stage simulated annealing (MSA) algorithm (see Algorithm 2). The number of backup computing resource allocations obtained in this step is denoted by $\boldsymbol{\Theta}$, which is a given parameter. Then this work removes the infeasible backup computing allocations from $\boldsymbol{F}$ with considering (4.12e). This work sorts the remaining backup computing resource allocations in $\boldsymbol{F}$ in a nondecreasing order of the required backup computing capacity. The DBA algorithm passes through the backup computing

resource allocations in this order; when it comes to one backup computing resource allocation, this work allocates the backup transmission resource for backup paths of virtual links by using a greedy sharing (GS) algorithm (see Algorithm 3). If we obtain a feasible solution with backup transmission resource allocation, the algorithm is terminated; otherwise, it goes to the next backup computing resource allocation in F.

Algorithm 1: Disjoint backup allocating (DBA)

Input: p, ϵ, Θ, T^{init}, ρ, and primary computing and transmission resource allocation

Ouput: Backup computing and transmission resource allocation and required backup computing capacity

Generate set of backup computing resource allocations of F by using Algorithm 2

Remove infeasible allocations from F

Sort remaining allocations in F

for Backup computing resource allocation in F **do**

  Compute backup transmission resource allocation by using Algorithm 3

  if Obtain feasible solution **then**

    Break

  end

end

4.3.2 Backup computing resource allocation

This work introduces the MSA algorithm, which is extended from a typical SA algorithm [91]. Consider T^{init} and $0 < \rho < 1$ as two given parameters in the typical SA algorithm. The running variable indicating the "temperature" in the typical SA algorithm is represented by T. The MSA algorithm divides the typical SA into Θ stages. Let given parameter T_a represent the terminal "temperature" of stage $a \in [1, \Theta]$; $T_1 < T^{\text{init}}$ and $T_a > T_{a'}$ if $a < a'$. At the beginning of MSA algorithm, $T = T^{\text{init}}$ is set. Given the values of p and ϵ, this work prepares a table whose gth entry denotes the value of $\Gamma_{n^{\text{B}}}$ when $|L_{n^{\text{B}}}| = g$ based on (4.2). The required backup computing capacity on

each backup facility node is computed by using the table and (4.3). Then the MSA algorithm goes through the stages from 1 to Θ. At the beginning of each stage, an initial backup computing resource allocation is generated by allocating the backup computing resources for virtual nodes in a random order. For a virtual node, this work allocates its backup computing resource on the backup facility node such that the allocation can maximize the ratio of remaining capacity after the allocation to maximum capacity of the backup facility node. The remaining capacity after an allocation equals to the value obtained by subtracting the additional required backup computing capacity from the current remaining capacity, which is negative when (4.12e) is not satisfied. Each stage may contain several iterations. In each iteration, $T = \rho T$; a new allocation is generated by reallocating a random virtual node based on the existing allocation. If the total required backup computing capacity of new allocation is less than that of the existing one, the new allocation is accepted and replaces the existing one; otherwise, the new allocation is accepted with a probability, which depends on the value of T and the difference between the required backup computing capacities of new and existing allocations. The algorithm outputs the existing allocation, which is amended to F, and moves to stage $a + 1$ when $T \leq T_a$.

4.3.3 Backup transmission resource allocation

Overall of GS

In the GS algorithm, this work allocates each backup path one by one based on a backup computing resource allocation. For a backup path, this work obtains the set of all allocated paths which are mutually exclusive with it based on the analysis in Section 4.2.3. Let $V^{\mathrm{B}}_{lv^{\mathrm{B}}}$ and $V^{\mathrm{P}}_{lv^{\mathrm{B}}}$ denote sets of backup and primary paths, respectively, each of which has been allocated in substrate link $l \in L^{\mathrm{S}}$ and is mutually exclusive with backup path $v^{\mathrm{B}} \in V^{\mathrm{B}}$. By using the algorithms introduced in Seciton 4.3.3, this work computes the required backup transmission capacity and the marginal gain on each substrate link if backup path v^{B} is added to it. This work removes each substrate link where adding backup path v^{B} exceeds its maximum capacity. Based on the amended network, this work allocates backup path v^{B} to the substrate path with the smallest total marginal gains by using Dijkstra's algorithm, where a substrate

Algorithm 2: Multiple-stage simulated annealing (MSA)

Input: p, ϵ, Θ, T^{init}, ρ, and primary computing resource allocation
Ouput: F
Set $F = \emptyset$
Set $T = T^{\text{init}}$
Prepare table of $\Gamma_{n^{\text{B}}}$
for $a \in [1, \Theta]$ **do**
  Generate allocation by allocating backup computing resources of virtual nodes in random order
  Compute total required backup computing capacity of c_{B}
  while $T > T_a$ **do**
    Set $T = \rho T$
    Generate new allocation by reallocating random virtual node
    Compute new total required backup computing capacity of c'_{B}
    Accept new feasible allocation with a probability of $\min(1, \delta)$, where $\delta = e^{(\frac{c_{\text{B}} - c'_{\text{B}}}{T})}$
  end
  Append existing allocation to F
end

link weight is set to the marginal gain for the link.

Algorithm to compute required backup transmission capacity

For substrate link $l \in L^{\text{S}}$, this work introduces Algorithm 4 to compute its required backup transmission capacity when backup path $v^{\text{B}} \in V^{\text{B}}$ is added to it based on the analyses in Sections 4.2.2, 4.2.3, and 4.2.3. b_l^{B} is the required backup transmission capacity on substrate link l before allocating backup path v^{B}. Let $\mathbb{W}^{\text{B}}_{lv^{\text{B}}\prime}$ represent the set of all maximal unexclusive subsets of backup paths allocated in substrate link l, where each maximal unexclusive set contains backup path $v^{\text{B}\prime}$.

In Algorithm 4, this work first obtains $\mathbb{W}^{\text{B}}_{lv^{\text{B}}}$ with considering that backup path v^{B} is allocated by using the collected data before allocating backup path v^{B}, such as $\mathbb{W}^{\text{B}}_{lv^{\text{B}}\prime}, \forall v^{\text{B}\prime} \in \Delta^{\text{B}}_{lv^{\text{B}}} = V^{\text{B}}_l \setminus V^{\text{B}}_{lv^{\text{B}}}$; the details of this step are

Algorithm 3: Greedy sharing (GS)

Input: Backup computing resource allocation
Ouput: Backup transmission resource allocation
for VN $m \in M$ **do**
  for virtual link $k \in L_m^{\mathrm{D}}$ **do**
    for backup path v^{B} in set of three backup paths of k **do**
      Obtain all allocated paths mutually exclusive with backup path v^{B}
      Compute required backup transmission capacity and marginal gain on each substrate link
      Remove substrate links with insufficient remaining capacity
      Select substrate path with smallest total marginal gains
      Update required backup transmission capacity
    end
  end
end

Algorithm 4: Compute required backup transmission capacity

Input: b_l^{B}, $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}, \forall v^{\mathrm{B}\prime} \in \Delta_{lv^{\mathrm{B}}}^{\mathrm{B}} = V_l^{\mathrm{B}} \setminus V_{lv^{\mathrm{B}}}^{\mathrm{B}}$, $V_{lv^{\mathrm{B}}}^{\mathrm{P}}$, and $V_{lv^{\mathrm{B}}}^{\mathrm{P}}, \forall v^{\mathrm{B}\prime} \in \Delta_{lv^{\mathrm{B}}}^{\mathrm{B}}$
Ouput: κ
Obtain $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$ by Algorithm 5
Compute $b_{lv^{\mathrm{B}}}^{\mathrm{B}}$ by Algorithm 6
Set $\kappa = \max(b_l^{\mathrm{B}}, b_{lv^{\mathrm{B}}}^{\mathrm{B}})$

described as Algorithm 5. Then this work computes the maximum required backup transmission capacity among the maximal unexclusive sets in $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$, which is denoted as $b_{lv^{\mathrm{B}}}^{\mathrm{B}}$, by using Algorithm 6. Let κ denote the required backup transmission capacity on substrate link l after adding backup path v^{B}, which is the greater one between b_l^{B} and $b_{lv^{\mathrm{B}}}^{\mathrm{B}}$.

The basic idea of Algorithm 5 is to generate $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$ based on $\{v^{\mathrm{B}}\}$ and $\Delta_{lv^{\mathrm{B}}}^{\mathrm{B}}$ according to $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}, \forall v^{\mathrm{B}\prime} \in \Delta_{lv^{\mathrm{B}}}^{\mathrm{B}}$. If backup path v^{B} is mutually exclusive with each backup path allocated in substrate link l, or $V_{lv^{\mathrm{B}}}^{\mathrm{B}} = V_l^{\mathrm{B}}$ and $\Delta_{lv^{\mathrm{B}}}^{\mathrm{B}} = \emptyset$, $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$ only contains the set of $\{v^{\mathrm{B}}\}$. Otherwise, any maximal unexclusive set in

$\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}}$ is a union set of $\{v^{\mathrm{B}}\}$ and a subset of $\Delta^{\mathrm{B}}_{lv^{\mathrm{B}}}$; $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}}$ covers all backup paths in $\Delta^{\mathrm{B}}_{lv^{\mathrm{B}}}$. For the latter case, there are two situations to include backup path $v^{\mathrm{B}\prime} \in \Delta^{\mathrm{B}}_{lv^{\mathrm{B}}}$ into $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}}$. If there is at least one $W^{\mathrm{B}}_{l} \in \mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}\prime}}$ that is a subset of $\Delta^{\mathrm{B}}_{lv^{\mathrm{B}}}$, this work adds the union of $\{v^{\mathrm{B}}\}$ and each W^{B}_{l} as a maximal unexclusive set in $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}}$; otherwise, this work adds $\{v^{\mathrm{B}}\} \cup \{v^{\mathrm{B}\prime}\}$ as a maximal unexclusive set in $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}}$.

Algorithm 5: Find all maximal unexclusive sets

Input: $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}\prime}}, \forall v^{\mathrm{B}\prime} \in \Delta^{\mathrm{B}}_{lv^{\mathrm{B}}}$
Ouput: $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}}$
Set $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}} = \emptyset$
if $\Delta^{\mathrm{B}}_{lv^{\mathrm{B}}} = \emptyset$ **then**
  Set $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}} = \{\{v^{\mathrm{B}}\}\}$
else
  for $v^{\mathrm{B}\prime} \in \Delta^{\mathrm{B}}_{lv^{\mathrm{B}}}$ **do**
    Set $Flag = 0$
    for $W^{\mathrm{B}}_{l} \in \mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}\prime}}$ **do**
      if $W^{\mathrm{B}}_{l} \subseteq \Delta^{\mathrm{B}}_{lv^{\mathrm{B}}}$ **then**
        Set $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}} \leftarrow \mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}} \cup \{\{v^{\mathrm{B}}\} \cup W^{\mathrm{B}}_{l}\}$
        Set $Flag = 1$
      end
    end
    if $Flag = 0$ **then**
      Set $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}} \leftarrow \mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}} \cup \{\{v^{\mathrm{B}}\} \cup \{v^{\mathrm{B}\prime}\}\}$
    end
  end
end

This work computes the maximum required backup transmission capacity among maximal unexclusive sets in $\mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}}$ in Algorithm 6. If $V^{\mathrm{P}}_{lv^{\mathrm{B}}} = \emptyset$, the required backup transmission capacity of $W^{\mathrm{B}}_{l} \in \mathbb{W}^{\mathrm{B}}_{lv^{\mathrm{B}}}$ is the sum of demanded capacity of backup path v^{B} and required backup transmission capacity of $W^{\mathrm{B}\prime}_{l} = W^{\mathrm{B}}_{l} \setminus \{v^{\mathrm{B}}\}$, which has been collected. Otherwise, this work constructs a directed graph based on W^{B}_{l} and W^{P}_{l} and use the algorithm presented in [103]

solving the MCF problem to compute the required backup transmission capacity of W_l^{B}, where $W_l^{\mathrm{P}} = \cup_{v^{\mathrm{B}\prime} \in W_l^{\mathrm{B}}} V_{lv^{\mathrm{B}\prime}}^{\mathrm{P}}$ is obtained by the collected data of $V_{lv^{\mathrm{B}\prime}}^{\mathrm{P}}, v^{\mathrm{B}\prime} \in W_l^{\mathrm{B}}$.

Algorithm 6: Compute $b_{lv^{\mathrm{B}}}^{\mathrm{B}}$

Input: $V_{lv^{\mathrm{B}}}^{\mathrm{P}}$, $V_{lv^{\mathrm{B}\prime}}^{\mathrm{P}}, \forall v^{\mathrm{B}\prime} \in \Delta_{lv^{\mathrm{B}}}^{\mathrm{B}}$, and $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$
Ouput: $b_{lv^{\mathrm{B}}}^{\mathrm{B}}$
Set $b_{lv^{\mathrm{B}}}^{\mathrm{B}} = 0$
for $W_l^{\mathrm{B}} \in \mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$ **do**
  if $V_{lv^{\mathrm{B}}}^{\mathrm{P}} = \emptyset$ **then**
    Set $b_{W_l^{\mathrm{B}}}^{\mathrm{B}} = d_{v^{\mathrm{B}}} + b_{W_l^{\mathrm{B}\prime}}^{\mathrm{B}}$, where $W_l^{\mathrm{B}\prime} = W_l^{\mathrm{B}} \setminus \{v^{\mathrm{B}}\}$
  else
    Obtain $W_l^{\mathrm{P}} = \cup_{v^{\mathrm{B}\prime} \in W_l^{\mathrm{B}}} V_{lv^{\mathrm{B}\prime}}^{\mathrm{P}}$
    Construct directed graph
    Obtain $b_{W_l^{\mathrm{B}}}^{\mathrm{B}}$ by solving MCF problem
  end
  Set $b_{lv^{\mathrm{B}}}^{\mathrm{B}} \leftarrow \max(b_{lv^{\mathrm{B}}}^{\mathrm{B}}, b_{W_l^{\mathrm{B}}}^{\mathrm{B}})$
end

This work shows an example to demonstrate Algorithms 4-6. Consider adding backup path $v^{\mathrm{B}} \in V^{\mathrm{B}}$ to substrate link $l \in L^{\mathrm{S}}$, where $V_l^{\mathrm{B}} = \{v_1^{\mathrm{B}}, v_2^{\mathrm{B}}, v_3^{\mathrm{B}}, v_4^{\mathrm{B}}\}$ and $V_l^{\mathrm{P}} = \{v_1^{\mathrm{P}}, v_2^{\mathrm{P}}, v_3^{\mathrm{P}}, v_4^{\mathrm{P}}\}$. We are given b_l^{B}, $V_{lv^{\mathrm{B}}}^{\mathrm{B}} = \{v_4^{\mathrm{B}}\}$ and $\Delta_{lv^{\mathrm{B}}}^{\mathrm{B}} = \{v_1^{\mathrm{B}}, v_2^{\mathrm{B}}, v_3^{\mathrm{B}}\}$, $V_{lv^{\mathrm{B}}}^{\mathrm{P}} = \{v_1^{\mathrm{P}}, v_4^{\mathrm{P}}\}$, $\mathbb{W}_{lv_1^{\mathrm{B}}} = \{\{v_1^{\mathrm{B}}, v_2^{\mathrm{B}}\}\}$, $\mathbb{W}_{lv_2^{\mathrm{B}}} = \{\{v_1^{\mathrm{B}}, v_2^{\mathrm{B}}\}\}$, $\mathbb{W}_{lv_3^{\mathrm{B}}} = \{\{v_3^{\mathrm{B}}, v_4^{\mathrm{B}}\}\}$, and $V_{lv^{\mathrm{B}\prime}}^{\mathrm{P}} = \{v_1^{\mathrm{P}}, v_2^{\mathrm{P}}\}, \forall v^{\mathrm{B}\prime} \in \Delta_{lv^{\mathrm{B}}}^{\mathrm{B}}$.

In Algorithm 5, for backup paths v_1^{B} and v_2^{B}, since $\{v_1^{\mathrm{B}}, v_2^{\mathrm{B}}\} \subseteq \Delta_{lv^{\mathrm{B}}}^{\mathrm{B}}$, we add $\{v_1^{\mathrm{B}}, v_2^{\mathrm{B}}, v^{\mathrm{B}}\}$ in $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$; for backup path v_3^{B}, since $\{v_3^{\mathrm{B}}, v_4^{\mathrm{B}}\} \nsubseteq \Delta_{lv^{\mathrm{B}}}^{\mathrm{B}}$, we add $\{v_3^{\mathrm{B}}, v^{\mathrm{B}}\}$ in $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$. As a result, we obtain $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}} = \{\{v_1^{\mathrm{B}}, v_2^{\mathrm{B}}, v^{\mathrm{B}}\}, \{v_3^{\mathrm{B}}, v^{\mathrm{B}}\}\}$. In Algorithm 6, since $V_{lv^{\mathrm{B}}}^{\mathrm{P}}$ is not empty, we solve the MCF problem for each $W_l^{\mathrm{B}} \in \mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$, where $W_l^{\mathrm{P}} = \{v_1^{\mathrm{P}}, v_2^{\mathrm{P}}, v_4^{\mathrm{P}}\}$, and obtain $b_{lv^{\mathrm{B}}}^{\mathrm{B}}$. Finally, we obtain κ by comparing b_l^{B} and $b_{lv^{\mathrm{B}}}^{\mathrm{B}}$.

4.3.4 Computational time complexity of DBA

There are at most $|M|$ virtual nodes on a backup facility node. To prepare the table of $\Gamma_{n^{\mathrm{B}}}$, we need to compute the value of $\Gamma_{n^{\mathrm{B}}}$ for each value of $|L_{n^{\mathrm{B}}}| \in [0, |M|]$. Given $|L_{n^{\mathrm{B}}}|$, it takes $O(|L_{n^{\mathrm{B}}}|^2)$ to obtain the value of $\Gamma_{n^{\mathrm{B}}}$ by computing (4.2). Therefore, the computational complexity of preparing the table is $O(|M|^3)$.

To compute the required backup computing capacity on a backup facility node, this work sorts the protected virtual nodes by their demanded computing capacities, which takes $O(|M| \log |M|)$. Then it takes $O(|M|)$ to compute (4.3). Therefore, allocating the backup computing resource of a virtual node to a backup facility node takes $O(|N_{\mathrm{B}}^{\mathrm{S}}||M| \log |M|)$. It leads to $O(|N_{\mathrm{B}}^{\mathrm{S}}||M| \log |M| \sum_{m \in M} |N_m^{\mathrm{D}}|)$ to generate a backup computing resource allocation and to compute the total required backup computing capacity at the beginning of each stage of the MSA algorithm. Similarly, for each iteration in each stage, generating a new allocation with computing the total required backup computing capacity takes $O(|N_{\mathrm{B}}^{\mathrm{S}}||M| \log |M|)$. Considering that there are $O(\Theta)$ stages, the computational time complexity of MSA algorithm is $O(\iota)$, where $\iota = |M|^3 + |N_{\mathrm{B}}^{\mathrm{S}}||M| \log |M| \sum_{m \in M} |N_m^{\mathrm{D}}|\Theta$. It takes $O(|N_{\mathrm{B}}^{\mathrm{S}}|\Theta)$ to remove all infeasible allocations. Sorting the Θ backup computing resource allocations takes $O(\Theta \log \Theta)$.

Let $\Psi = \sum_{m \in M} |L_m^{\mathrm{D}}|$. Consider backup path $v^{\mathrm{B}} \in V^{\mathrm{B}}$ as the gth backup path to be allocated, where $g \in [1, 3\Psi]$. The number of prepared paths which have been allocated before backup path v^{B} is $\Psi + g - 1$. Obtaining all allocated paths which are mutually exclusive with backup path v^{B} needs $O(\Psi + g - 1)$, which means that $O(\Psi^2)$ is required for all backup paths.

There are at most $g - 1$ allocated backup paths on a substrate link, which lead to $O(3^{\frac{g}{3}})$ maximal unexclusive subsets based on Theorem 3. Determining whether a maximal unexclusive set is a subset of $\Delta_{lv^{\mathrm{B}}}^{\mathrm{B}}$ takes $O(\Psi)$. Therefore, obtaining $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$ by Algorithm 5 takes $O(\Psi 3^{\frac{g}{3}})$. Similarly, there are $O(3^{\frac{g}{3}})$ maximal unexclusive subsets in $\mathbb{W}_{lv^{\mathrm{B}}}^{\mathrm{B}}$. Obtaining W_l^{P} and constructing the directed graph take $O(\Psi^2)$. Solving the MCF problem takes $O(\Psi^4 \log \Psi)$. Hence, running Algorithm 6 to compute $b_{lv^{\mathrm{B}}}^{\mathrm{B}}$ takes $O(3^{\frac{g}{3}} \Psi^4 \log \Psi)$, which is the computational complexity of Algorithm 4. $O(3^{\frac{g}{3}} \Psi^4 |L^{\mathrm{S}}| \log \Psi)$ and $O(3^{\Psi} \Psi^4 |L^{\mathrm{S}}| \log \Psi)$

are required for backup path v^{B} and all backup paths, respectively, to compute the required backup transmission capacities on all substrate links.

It takes $O(\Psi|L^{\mathrm{S}}|\log|N^{\mathrm{S}}|)$ to select the substrate paths with minimum total marginal gains for all backup paths by using Dijkstra's algorithm. As a result, the computational time complexity of GS algorithm to allocate the backup transmission resource is $O(3^{\Psi}\Psi^4|L^{\mathrm{S}}|\log\Psi)$.

Therefore, the the computational time complexity of DBA to solve P-FTS is the dominant term of $O(\Theta 3^{\Psi}\Psi^4|L^{\mathrm{S}}|\log\Psi)$. The term of $O(3^{\Psi})$ is the worst-case computation time for finding all maximal unexclusive sets, which is similar to listing all maximal cliques in a clique problem [104]. By adjusting the procedure of computing the required backup transmission capacity, DBA is modified to solve P-LTS and P-NTS. Then the computational time complexity for each of the two cases is the dominant term of $O(\iota + \Theta\Psi|L^{\mathrm{S}}|\log|N^{\mathrm{S}}|)$.

4.4 Numerical results

This work considers a non-sharing (NS) model as a baseline, where the dedicated protection is provided for the computing resource, or $\epsilon = 0$; cross-sharing and backup-sharing are not considered for the backup transmission resource allocation. The proposed model is considered with different degrees of transmission resource sharing, which leads to P-FTS, P-LTS, and P-NTS. In Section 4.4.1, the size of problem, such as the number of substrate nodes and links, is small, and this work solves NS and P-NTS by the MILP approach. For a large one in Section 4.4.2, this work solves them by the modified DBA heuristic algorithms. P-FTS and P-LTS are always solved by the DBA and modified DBA algorithms, respectively.

This work uses Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory for the evaluations. The MILP problem is solved by the IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer with version 12.8 [88].

4.4.1 Demonstration

This work demonstrates different models on the example provided in Fig. 4.1. This work sets $p = 10^{-2}$ and $\epsilon = 10^{-4}$. This work obtains that there is no feasible solution for NS. This is because NS requires the same amount

of computing resources with the primary allocation for backup, which exceeds the maximum capacity of a backup facility node. More specifically, the backup computing resource of virtual node j_6 needs to be allocated in the same backup facility node with that of one virtual node from VN m_1. Virtual node j_6 and any virtual node from VN m_1 demand 6 and 5 unites of backup computing capacity, respectively. However, each backup facility node has the maximum capacity less than 11. As a result, there is no feasible backup computing resource allocation for NS.

Figure 4.8 and Table 4.3 show the optimal backup computing and transmission resource allocations for P-NTS, respectively. This work obtains that in total 18 units of backup computing resource are required for P-NTS, which saves 7 units of backup computing resource compared to providing the dedicated protection.

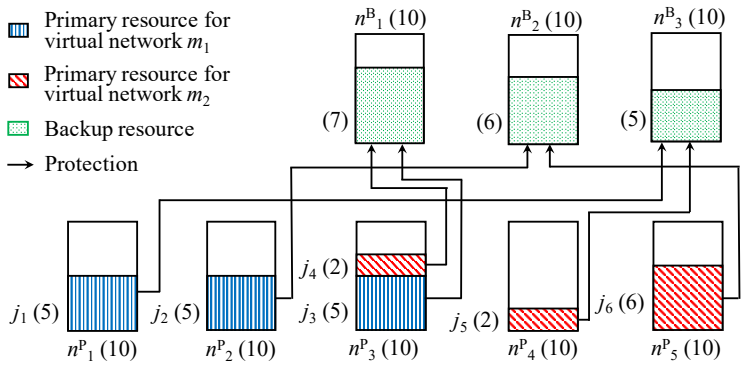


Figure 4.8: Backup computing resource allocation in optimal solution for P-NTS.

Table 4.3: Backup transmission resource allocation in optimal solution for P-NTS.

Virtual link	1st backup path	2nd backup path	3rd backup path
k_1	(l_8, l_5)	(l_4, l_1)	(l_8, l_6)
k_2	(l_6)	(l_2, l_1)	(l_3)
k_3	(l_{11}, l_{10}, l_9)	(l_2)	(l_2, l_1, l_5, l_8)
k_4	(l_3, l_6, l_9)	(l_8)	(l_3, l_7)
k_5	(l_{11})	(l_6, l_9)	(l_7)
k_6	$(l_2, l_1, l_5, l_9, l_{10})$	(l_4, l_5)	(l_3)

P-FTS and P-LTS have the same optimal solution. The backup computing

resource allocation in the optimal solution is the same with that shown in Fig. 4.2. Table 4.4 shows the corresponding backup transmission resource allocation. P-FTS and P-LTS require 16 units of backup computing resource; compared to providing the dedicated protection, 9 units of backup computing resource are saved by the proposed model with the probabilistic protection. Furthermore, compared to P-NTS, P-FTS and P-LTS save 2 units of backup computing resource. This is because cross-sharing and backup-sharing are considered for the backup transmission resource allocation, which reduces the required backup transmission capacity. More feasible solutions for backup computing resource allocation can exist by requiring less backup transmission capacity. As a result, the required backup computing capacity is reduced.

Table 4.4: Backup transmission resource allocation in optimal solution for P-FTS and P-LTS.

Virtual link	1st backup path	2nd backup path	3rd backup path
k_1	(l_1, l_2)	(l_2, l_3)	(l_3)
k_2	(l_5, l_8)	(l_6)	(l_6, l_8)
k_3	(l_6, l_3)	(l_2, l_3, l_7)	(l_3, l_7)
k_4	(l_3, l_6, l_9)	(l_8)	(l_3, l_7)
k_5	(l_9, l_6)	(l_{11})	(l_6, l_8)
k_6	(l_{11}, l_7, l_3)	(l_6)	(l_3)

For example, based on Table 4.4, several backup paths of virtual links k_1, k_3, k_4, and k_6 are routed through substrate link l_3; in total 11 units of backup transmission resource are required on substrate link l_3 when P-NTS is adopted, which exceeds its maximum capacity of 6. This work shows that the capacity constraint is satisfied by using any of P-FTS and P-LTS. Considering backup-sharing among backup paths of the same virtual link, the backup transmission resource is required to be reserved for at most one backup path for each virtual link. According to the analyses in Section 4.2.3, the second backup paths of virtual links k_1 and k_3, the first backup path of virtual link k_4, and the third backup path of virtual link k_6 are not mutually exclusive. Therefore, without considering cross-sharing, in total 5 units of backup transmission resource are required. Since the primary path of virtual link k_3 is mutually exclusive with any of the four backup paths, 1 unit of backup transmission resource is saved by cross-sharing. Therefore, only 4 units of backup transmission resource are required on substrate link l_3 in P-FTS and P-LTS.

4.4.2 Evaluation

Experiment setup

A 24-node undirected US backbone network presented in [105] is used as the substrate network, where $|N_{\mathrm{P}}^{\mathrm{S}}|$ and $|N_{\mathrm{B}}^{\mathrm{S}}|$ are set to 18 and 6, respectively. This work considers each VN as a full meshed network. For each result, this work conducts 500 trials to obtain the average value. The parameters for each trial are set as follows. This work randomly decides the locations of primary and backup facility nodes. The number of VNs and the number of virtual nodes in each VN are randomly selected from the ranges of [2, 10] and [2, 4], respectively. The demanding computing and transmission capacities of each virtual node and virtual link are randomly set within the range of [1, 10]. The capacity constraints are not considered to generate the primary resource allocation. Each virtual node is randomly embedded in a primary facility node; the primary path of each virtual link is mapped to the path connecting two virtual nodes with the minimum number of substrate links. After the primary resource allocation, the remaining computing and transmission capacities on each substrate node and link are randomly set within the ranges of $[0, A]$ and $[0, B]$, respectively, where A and B denote the upper bounds of ranges.

This work compares the different models in terms of feasibility, required backup computing capacity, and computation time. The feasibility of a model, which is denoted by ξ, is defined as the ratio of number of trials that returns a feasible solution to the total number of trials. Note that only the trials with feasible solutions for all the models are included to compute the average required backup computing capacity and average computation time for each model. Let η denote a ratio of the required backup computing capacity in a feasible solution to the total primary computing capacity. Clearly, for NS, $\eta_{\mathrm{NS}} = 1$. The performance dependencies on the values of p and ϵ and on the capacities of substrate nodes and links are evaluated.

Dependencies on values of p and ϵ

Table 4.5 shows the performance dependencies on value of ϵ for different models, where $p = 10^{-2}$ and $A = B = 100$. P-NTS requires less backup computing capacity than NS; the feasibilities of NS and P-NTS are comparable. It indi-

cates that the main bottleneck to get a feasible solution is the transmission capacity on substrate links. By considering limited backup transmission resource sharing, P-LTS achieves the feasibility of 0.87 in average, which is 2.38 times greater than those of NS and P-NTS. P-FTS further improves the the average feasibility to 0.90 by minimizing the required backup transmission capacity. Compared to NS, P-NTS, P-LTS, and P-FTS save in average 53%, 54%, and 54% required backup computing capacity, respectively. From Table 4.5, this work observes that as the value of ϵ increases, the required backup computing capacity of each of P-NTS, P-LTS, and P-FTS decreases. This is because, for the probabilistic protection, as the value of ϵ increases, the backup computing resource sharing is more acceptable, which decreases the required backup computing capacity.

Table 4.5: Dependencies on value of ϵ for different models.

ϵ	ξ_{NS}	$\xi_{\text{P-NTS}}$	$\xi_{\text{P-LTS}}$	$\xi_{\text{P-FTS}}$	$\eta_{\text{P-NTS}}$	$\eta_{\text{P-LTS}}$	$\eta_{\text{P-FTS}}$
10^{-4}	0.38	0.37	0.88	0.91	0.59	0.57	0.56
5×10^{-4}	0.39	0.37	0.86	0.86	0.48	0.47	0.47
10^{-3}	0.37	0.34	0.89	0.93	0.46	0.45	0.45
5×10^{-3}	0.39	0.37	0.87	0.89	0.42	0.41	0.40
7.5×10^{-3}	0.40	0.38	0.86	0.91	0.42	0.41	0.40

Table 4.6 shows the performance dependencies on value of p for different models, where $\epsilon = 10^{-4}$ and $A = B = 100$. As the value of p increases, more backup computing resources are required to guarantee the same degree of probabilistic protection, which indicates that the required backup computing capacities of P-NTS, P-LTS, and P-FTS increase. Tables 4.5 and 4.6 observe that the feasibilities of each model are comparable for different values of p and ϵ. It indicates that, in the examined cases, the feasibility is mainly limited by the transmission capacity instead of the computing capacity.

Table 4.6: Dependencies on value of p for different models.

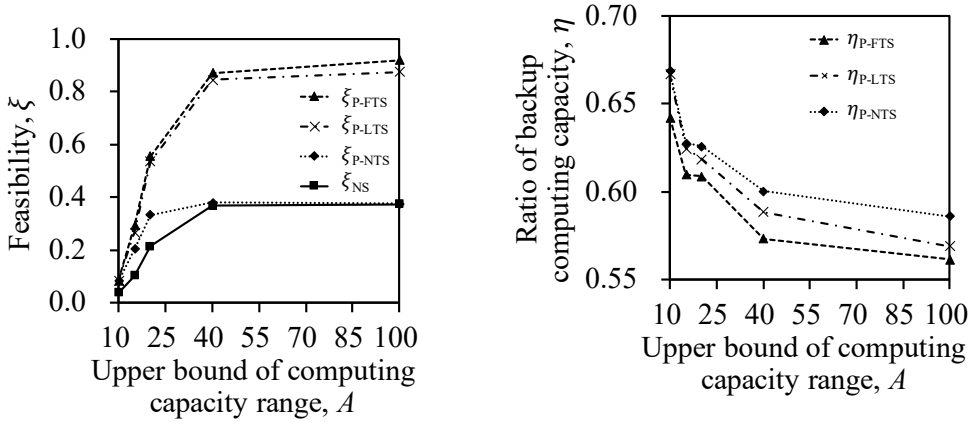
p	ξ_{NS}	$\xi_{\text{P-NTS}}$	$\xi_{\text{P-LTS}}$	$\xi_{\text{P-FTS}}$	$\eta_{\text{P-NTS}}$	$\eta_{\text{P-LTS}}$	$\eta_{\text{P-FTS}}$
5×10^{-4}	0.41	0.39	0.87	0.90	0.44	0.43	0.43
10^{-3}	0.38	0.36	0.84	0.90	0.45	0.44	0.44
5×10^{-3}	0.39	0.38	0.86	0.93	0.49	0.47	0.46
10^{-2}	0.38	0.37	0.88	0.91	0.59	0.57	0.56
10^{-1}	0.38	0.37	0.88	0.90	0.97	0.97	0.96

Dependencies on capacities of substrate nodes and links

Figure 4.9 shows the performance dependencies on the computing capacity of a substrate node for different models, where $p = 10^{-2}$, $\epsilon = 10^{-4}$, and $B = 100$. Figure 4.9(a) observes that when the value of A is small, or $A = 10$, since the computing capacity is a main bottleneck for each model to obtain a feasible solution, the feasibilities of different models are low and comparable. As the value of A increases from 10 to 40, the feasibility of each model significantly increases. Since P-FTS and P-LTS consider both backup computing and transmission resource sharing, they outperform the other two models; P-FTS outperforms P-LTS by minimizing the required backup transmission capacity. P-NTS has a higher feasibility than NS by requiring much less backup computing capacity. As the value of A increases from 40, the computing capacity becomes sufficient and the transmission capacity becomes the main bottleneck. As a result, the feasibility of each model slightly increases; the feasibilities of NS and P-NTS become comparable. From Fig. 4.9(b), this work observes that the required backup computing capacity decreases for each of P-NTS, P-LTS, and P-FTS as the value of A increases. This is because, when the computing capacity on each substrate node is small, the backup computing resource allocations for virtual nodes need to be distributed over different substrate nodes, which limits the backup computing resource sharing among virtual nodes. As the value of A increases, the allocations can be more centralized to promote the backup computing resource sharing. Consequently, the required backup computing capacity decreases.

Figure 4.10 shows the performance dependencies on the transmission capacity of a substrate link for different models, where $p = 10^{-2}$, $\epsilon = 10^{-4}$, and $A = 100$. Since transmission capacity is the main bottleneck to obtain a feasible solution, Fig. 4.10(a) shows that the feasibility of each model increases as the value of B increases. P-FTS provides the highest feasibility; especially, for $B = 20$, the feasibility of P-FTS is 2.08 times greater than that of P-LTS.

Figure 4.10(b) observes that the required backup computing capacity decreases for each of P-NTS, P-LTS, and P-FTS as the value of B increases. This is because more transmission capacity in each substrate link leads to more feasible solutions for backup computing resource allocation. Furthermore, this

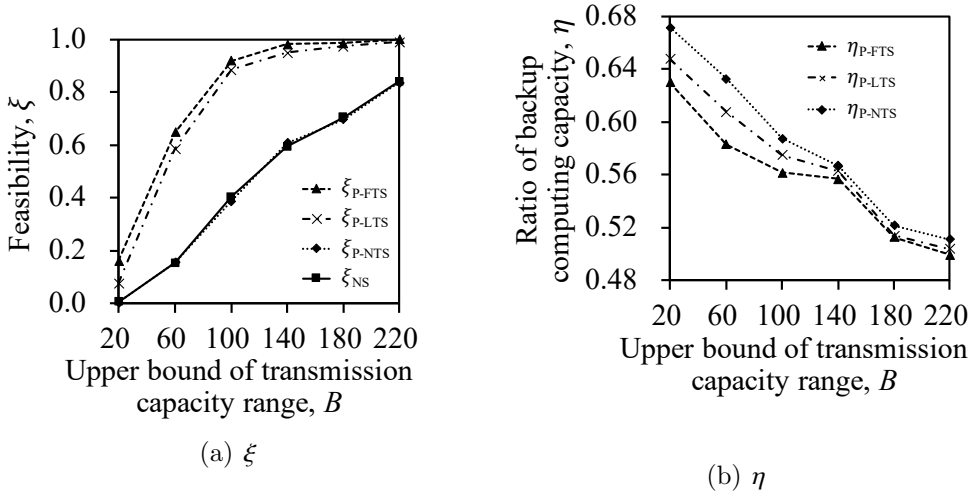


(a) ξ

(b) η

Figure 4.9: Dependency on capacity of substrate node.

work observes that when the transmission capacity is sufficient, or $\boldsymbol{B} = 220$, the required computing capacities of P-NTS, P-LTS, and P-FTS are comparable. When the transmission capacity is limited, or $\boldsymbol{B} = 20$, P-FTS significantly outperforms P-NTS and P-LTS by considering full backup transmission resource sharing.



(a) ξ

(b) η

Figure 4.10: Dependency on capacity of substrate link.

Computation time

Table 4.7 shows the computation time [s] to obtain the results shown in Fig. 4.10. This work observes that NS and P-FTS require the shortest and longest computation times, respectively; P-NTS requires slightly shorter time than P-LTS. Especially, the computation times of P-FTS in cases of $B = 180$ and $B = 220$ are much longer than those in other cases and those of other models. The computation time for each model increases as the value of B increases. This is because a value of computation time is the average value from trials that each model returns a feasible solution. When the value of B is small, only the trials where the problem sizes, such as the numbers of VNs, virtual nodes, and virtual links, are small return feasible solutions for each model. As a result, the computation time is short in each model. As the value of B increases, more trials that correspond to large size problems return feasible solutions, and then the computation time increases for each model, especially for P-FTS.

Figure 4.10 and Table 4.7 indicate that a network operator can set an appropriate degree of backup transmission resource sharing based on practical requirements, such as the admissible computation time. Typically, P-FTS is beneficial in terms of ξ and η in the cases with limited transmission capacity, where the computation time is acceptable. When the transmission capacity is sufficient, P-LTS can be considered to save the computation time with a slight performance degradation. P-NTS with much lower feasibility may be adopted to further reduce the computation time.

Table 4.7: Computation time [s] to obtain results shown in Fig. 4.10.

B	NS	P-NTS	P-LTS	P-FTS
20	0.009	0.014	0.016	0.065
60	0.012	0.015	0.017	0.791
100	0.013	0.018	0.018	6.934
140	0.015	0.021	0.022	4.467
180	0.017	0.023	0.024	115.793
220	0.017	0.024	0.026	147.572

4.5 Chapter summary

This chapter proposed a backup computing and transmission resource allocation model for VNs with the probabilistic protection against multiple facility node failures. Both backup computing and transmission resource allocations are considered to minimize the required backup computing capacity. Considering that the required backup transmission capacity can affect the required backup computing capacity, this work analyzed backup transmission resource sharing with multiple facility node failures based on graph theory. A heuristic algorithm was introduced to solve the problem. The result revealed that the proposed model outperforms the baseline in terms of both feasibility and required backup computing capacity. This work discussed the application scenarios for the proposed model with different degrees of backup transmission resource sharing. With the analyses, a network operator can consider an appropriate degree of backup transmission resource sharing based on practical requirements.

Chapter 5

Backup resource allocation model for network functions

This thesis proposes a backup resource allocation model for middleboxes with considering the importance of functions and both failure probabilities of functions and backup servers [48, 49]. In this model, a backup server is allowed to protect several functions; a function can have multiple backup servers. This work defines a weighted unavailability for each function, which depends on its importance and failure probability, the assignment of backup servers, and the failure probabilities of assigned backup servers. In this thesis, this work focuses on the assignment of backup servers to functions where the weighted unavailability of function that is in the worst case is minimized.

This work formulates the proposed backup resource allocation model as an MILP problem. This work analyzes the considered problem with proving that it is NP-complete and with showing how it is different with some similar classical problems, which indicates that developing new approaches with theoretical analyses to solve the problem is necessary and challenging.

This work introduces three heuristic algorithms with polynomial time complexity to solve the problem. This work analyzes the approximation performances of different heuristic algorithms by providing several lower and upper bounds. In numerical analysis, this work compares the three heuristic algorithms in terms of the deviation from the optimal value. This work evaluates the computation time used to solve the problem and compare different intro-

duced approaches in terms of computation time. The results show the pros and cons of different approaches. When the problem becomes large, solving the MILP problem needs a long computation time to obtain the optimal solution, but a relatively much shorter time to obtain a solution comparable to the optimal one. Heuristic approaches outperform the MILP approach when the admissible computation time is set short. The first two algorithms based on the greedy approach provide less deviations and require shorter computation time than the last heuristic based on the linear programming relaxation (LPR) approach. However, only the performance of last heuristic algorithm has an upper bound and there is no approximation guarantee for the first two. Referring to the analyses, a network operator can choose an appropriate approach according to the requirements in specific application scenarios.

The rest of this chapter is organized as follows. Section 5.1 presents the proposed optimization model. Section 5.2 describes two heuristic algorithms based on the greedy approach. Section 5.3 introduces another heuristic algorithm based on the LPR approach. The performances of different approaches are evaluated in Section 5.4. Section 5.5 summaries this chapter.

5.1 Model and problem definition

5.1.1 Assumptions

This model has several simplifying assumptions, which are often required for the analysis and are summarized as follows.

(i) All components of a VNF are deployed on one backup server to reach the ability of a hardware function.

(ii) A backup server can support several network functions at the same time.

(iii) Each network function requires the same amount of resources for information synchronization or recovery on a backup server.

(iv) All backup servers fail independently and each backup server fails independently of any function.

This work justifies some of the assumptions. A VNF can be composed of one or multiple internal components, each of which can be deployed over a virtual machine or container [106–108]. For assumptions (i) and (ii), we can improve the processing ability of a backup server by equipping with substantial computing resources or by adopting some advanced data processing frameworks [109].

5.1.2 Assign backup servers to protect functions

Let F and S represent a set of functions and a set of backup servers, respectively, where $|F|$ and $|S|$ denote the numbers of functions and backup servers, respectively. In general, for backup server $j \in S$, the information of c_j functions can be synchronized to it and at most c'_j functions can be recovered at the same time, where $c_j \geq c'_j$. Unless specifically stated, this work considers a special case of $c_j = c'_j$ to promptly recover each failed middlebox and the capacity of backup server $j \in S$ is represented by c_j in this thesis. Each function in F can be protected by several backup servers in S. Compared to an industry backup scheme where one or many dedicated backup servers are prepared for one specified function, the proposed model has overheads including that the number of logical connections between each backup server and functions is increased and that the configuration in a backup server becomes complicated.

This work considers multiple simultaneous failures among both functions and backup servers. When a function protected by some backup servers fails, one of the corresponding backup servers which does not fail promptly recovers the failed function by using the standby copy of its state. If a backup server does not fail, it can recover all the protected functions that fail simultaneously; otherwise, it cannot recover any failed function. The failure probabilities of function $i \in F$ and backup server $j \in S$ are considered as p_i and q_j, respectively.

The unavailability of function $i \in F$ is defined as the probability that function i becomes unavailable, which depends on the assignment of backup servers for function i. Let $x_{ij}, i \in F, j \in S$, represent a binary decision variable; x_{ij} is set to one if function $i \in F$ is protected by backup server $j \in S$ and zero otherwise. Consider F_j as a given parameter denoting a set of functions, each of which is prohibited to be protected by backup server $j \in S$, or $x_{ij} = 0, i \in F_j$.

Several constraints, such the transmission delay constraint with considering the placements of functions and backup servers, can be incorporated through F_j. For example, if the transmission delay between function $i \in F$ and backup server $j \in S$ is too high to satisfy the management requirements, function i is set to be contained in F_j in advance. There are two possible situations that a function becomes unavailable. One is that a function which is not protected by any backup server fails, and the other is that a function and all backup servers protecting it fail simultaneously. Therefore, the unavailability of function $i \in F$ is expressed by,

$$P_i = p_i \prod_{j \in S: x_{ij}=1} q_j. \tag{5.1}$$

Let $0 < w_i \leq 1$ represent the importance of function $i \in F$, which has been decided in advance and is considered as a given parameter. The function with greater value of w_i is more important than the one with smaller value of w_i. The dependency between functions in terms of importance is considered as that, for two functions with the same failure probability, the more important function can have smaller unavailability compared to the other by assigning backup servers. Considering w_i as the weight of function $i \in F$, the weighted unavailability of function i is expressed by,

$$P_i^{\mathrm{W}} = w_i p_i \prod_{j \in S: x_{ij}=1} q_j. \tag{5.2}$$

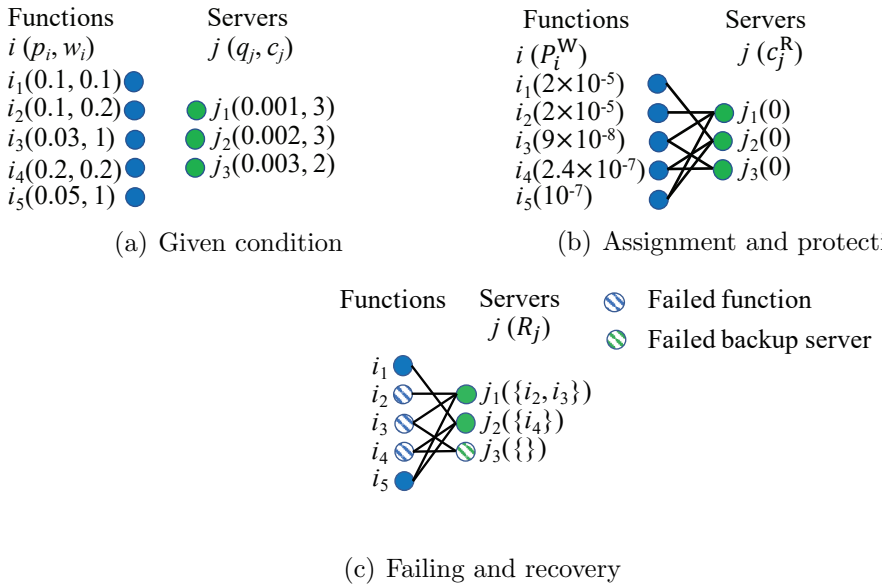
Equation (5.2) indicates that, for two functions $i, i' \in F$ with the same failure probability but different weights, or $p_i = p_{i'}$ and $w_i > w_{i'}$, in order to obtain the same weighted unavailability, function i with greater weight needs to be assigned with backup resources such that $\prod_{j \in S: x_{ij}=1} q_j < \prod_{j \in S: x_{i'j}=1} q_j$.

The worst weighted unavailability among $P_i^{\mathrm{W}}, i \in F$, which is denoted by Q, is expressed by,

$$Q = \max_{i \in F} P_i^{\mathrm{W}}. \tag{5.3}$$

Since an assignment can only exist between a function in F and a backup server in S, this work presents the assignments between functions in F and backup servers in S as a bipartite graph. Figure 5.1(a) shows an example of

given sets of functions and backup servers. Figure 5.1(b) depicts an example of assignment based on the given condition in Fig. 5.1(a), where c_j^{R} represents the remaining capacity in backup server $j \in S$ after the assignment. The unavailability of a function after the assignment can be obtained according to its weighted unavailability. Consider a situation that functions i_2, i_3, and i_4 and backup server j_3 fail simultaneously. Figure 5.1(c) shows a recovery example that failed functions i_2 and i_3 and failed function i_4 are recovered by surviving backup servers j_1 and j_2, respectively; R_j denotes the set of failed functions recovered by backup server $j \in S$.



(a) Given condition

(b) Assignment and protection

(c) Failing and recovery

Figure 5.1: Examples of protection, failing, and recovery.

5.1.3 Problem definition

The problem of backup resource allocation for middlebox with importance (BRAMI) is defined as follows:

Problem G iven a set of functions and a set of backup servers, both failure probabilities of functions and backup servers, importance of each function, capacity of each backup server, how to assign backup servers to functions to minimize the worst weighted unavailability, Q?

Note that, by incorporating the importance of functions into the BRAMI problem through (5.2), this work can handle the BRAMI problem in the same way as the problem without considering the importance; especially, each approach and its theorems introduced in this thesis hold for both problems.

5.1.4 Mixed integer linear programming problem

This work formulates the BRAMI problem as the following MILP problem.

$$\min \quad r \tag{5.4a}$$

$$s.t. \quad \sum_{i \in F} x_{ij} = c_j, \forall j \in S \tag{5.4b}$$

$$\log w_i + \log p_i + \sum_{j \in S} x_{ij} \log q_j \leq r, \forall i \in F \tag{5.4c}$$

$$x_{ij} = 0, \forall j \in S, i \in F_j \tag{5.4d}$$

$$x_{ij} \in \{0, 1\}, \forall i \in F, j \in S, \tag{5.4e}$$

where $e^r = Q$. The worst weighted unavailability, Q, is minimized in the objective function (5.4a) by minimizing r. Equation (5.4b) indicates that backup server $j \in S$ protects c_j functions. This work expresses (5.2) and (5.3) as a linear form in (5.4c) by taking the logarithmic for both sides of (5.2). Let Q^* be the worst weighted unavailability in an optimal solution obtained by solving the above MILP problem.

This thesis studies a static problem about how to assign a given set of backup servers to protect a given set of functions before any failure occurs. To consider the BRAMI problem with the dynamic scenarios, where functions and backup servers may fail or be recovered over time, an approach is to resolve the static BRAMI problem and re-allocate the backup resources once the given conditions change. A more practical approach may consider some other aspects, such as computational complexity, network bandwidth, and quality of service, related to the backup resource re-allocation.

5.1.5 NP-completeness

This work defines a backup resource allocation (BRA) decision problem as follows: given a set of functions and a set of backup servers, both failure

probabilities of functions and backup servers, capacity of each backup server, is it possible to find an assignment of backup servers to functions so that all the unavailabilities of functions are no more than u?

Theorem 4 *The BRA decision problem is NP-complete.*

Proof : The details of proof can be found in [37]. □

Based on Theorem 4, this work gives the following theorem for the BRAMI problem.

Theorem 5 *The BRAMI problem is NP-complete.*

Proof : Firstly, similar with the proof for Theorem 4, we can verify a certificate of any instance of the BRAMI problem in a polynomial time of $O(|F||S|)$ by computing (5.2) for $|F|$ times. Therefore, the BRAMI problem is NP.

By setting $w_i = 1, i \in F$, the BRAMI problem is the same with the BRA problem. In other words, the BRA problem is a subset of the BRAMI problem. Since the BRA decision problem is NP-complete, the BRAMI problem is also NP-complete. □

5.2 Greedy approach

The MILP problem introduced in (5.4a)-(5.4e) can be solved in a practical time, when the size of problem, such as the number of functions and backup servers, is small. However, for large one, it becomes intractable. In this section, this work considers two heuristic algorithms based on the greedy approach [57] to solve the BRAMI problem when its size becomes large.

5.2.1 Sorted greedy assignment

In a sorted greedy assignment (SGA) algorithm (see Algorithm 7), this work firstly sorts the backup servers in a nondecreasing order of failure probability and then make one pass through the backup servers in this order; when it comes to backup server $j \in S$, it assigns j to c_j functions with the greatest weighted

Algorithm 7: Sorted greedy assignment

Input: Importance and failure probability of function $i \in F$, w_i and p_i, capacity and failure probability of backup server $j \in S$, c_j and q_j

Ouput: Assignment and weighted unavailability of function $i \in F$, A_i and P_i^{W}

Set $P_i^{\mathrm{W}} = w_i p_i$ and $A_i = \emptyset$ for each function $i \in F$

Sort backup servers in a nondecreasing order of failure probability q_j

Assume that $q_1 \leq q_2 \leq \cdots \leq q_{|S|}$

for $j = 1, 2, \cdots, |S|$ **do**

  Let L be the set of c_j functions with the greatest weighted unavailabilities

  for $i \in L$ **do**

    Assign backup server j to function i

    Set $A_i \leftarrow A_i \cup \{j\}$

    Set $P_i^{\mathrm{W}} \leftarrow P_i^{\mathrm{W}} q_j$

  end

end

unavailabilities. In the beginning of algorithm, the weighted unavailability of function $i \in F$ is $w_i p_i$. Let A_i denote a set of backup servers assigned to function $i \in F$.

The SGA algorithm sorts backup servers for one time at the beginning, and sorts functions for $|S|$ times when it makes one pass through the sorted backup servers. The computational time complexities of sorting $|F|$ functions and sorting $|S|$ backup servers are $O(|F| \log |F|)$ and $O(|S| \log |S|)$, respectively. Therefore, the computational time complexity of SGA algorithm is $O(|S||F| \log |F|)$, where $|F| \geq |S|$ is considered.

Theorem 6 *The worst weighted unavailability in an optimal solution is at least* $\left(\prod_{i \in F} w_i p_i \prod_{j \in S} q_j^{c_j}\right)^{\frac{1}{|F|}}$, *or* $Q^* \geq \left(\prod_{i \in F} w_i p_i \prod_{j \in S} q_j^{c_j}\right)^{\frac{1}{|F|}}$.

Proof : Since Q^* is the worst weighted unavailability in the solution, or $Q^* = \max_{i \in F} P_i$, we obtain,

$$Q^{*|F|} \geq \prod_{i \in F} P_i = \prod_{i \in F} w_i p_i \prod_{j \in S} q_j^{c_j}. \tag{5.5}$$

Therefore, a lower bound on the optimum is obtained as,

$$Q^* \geq \left(\prod_{i \in F} w_i p_i \prod_{j \in S} q_j^{c_j} \right)^{\frac{1}{|F|}}. \tag{5.6}$$

□

5.2.2 Converse greedy assignment

This work introduces a converse greedy assignment (CGA) algorithm (see Algorithm 8), which is close to the SGA algorithm but performs the assignment in a converse way. In the CGA algorithm, this work firstly assigns each function with all the backup servers regardless of the capacity constraint of each backup server; the weighted unavailability of function $i \in F$ is $w_i p_i \prod_{j \in S} q_j$ after the assignment. As a result, there are $|F| - c_j$ functions exceeding the maximum capacity of backup server $j \in S$. Then this work sorts the backup servers in a nondecreasing order of failure probability and make one pass through the backup servers in this order; when it comes to backup server $j \in S$, this work withdraws the assignments between backup server j and $|F| - c_j$ functions with the smallest weighted unavailabilities. Clearly, the computational time complexity of CGA algorithm is $O(|S||F| \log |F|)$, which is the same with that of the SGA algorithm.

This work analyzes the approximation performance of CGA algorithm. Let Q^{CGA} represent the worst weighted unavailability obtained by the CGA algorithm.

Theorem 7 *When $w_i p_i = t, i \in F$, where t is a constant, and $c_j = |F| - 1, j \in S$, by using Algorithm 8, we can obtain a feasible assignment of backup servers to functions with weighted unavailability of function at most $\frac{Q^*}{\min_{j \in F} q_j}$, or $Q^{\mathrm{CGA}} \leq \frac{Q^*}{\min_{j \in F} q_j}$.*

Proof : Since $w_i p_i$ is a constant, all the functions are with the same weighted unavailability as $t \prod_{j \in S} q_j$ at the beginning of algorithm. For each backup server, we withdraw $|F| - c_j = 1$ assignment, where the involved function is with the smallest weighted unavailability. Let i' denote the function with worst weighted unavailability Q^{CGA} in the solution of CGA algorithm.

Algorithm 8: Converse greedy assignment

Input: Importance and failure probability of function $i \in F$, w_i and p_i, capacity and failure probability of backup server $j \in S$, c_j and q_j

Ouput: Assignment and weighted unavailability of function $i \in F$, A_i and P_i^{W}

Set $P_i^{\mathrm{W}} = w_i p_i \sum_{j \in S} q_j$ and $A_i = S$ for each function $i \in F$

Sort backup servers in a nondecreasing order of failure probability q_j

Assume that $q_1 \leq q_2 \leq \cdots \leq q_{|S|}$

for $j = 1, 2, \cdots, |S|$ **do**

  Let L' be the set of $|F| - c_j$ functions with the smallest weighted unavailabilities

  for $i \in L'$ **do**

    Withdraw the assignment from backup server j

    Set $A_i \leftarrow A_i \backslash \{j\}$

    Set $P_i^{\mathrm{W}} \leftarrow P_i^{\mathrm{W}} / q_j$

  end

end

Before obtaining the solution assignment, there must be at least one backup server withdrawn from function i'. Let j' be the last backup server withdrawn from function i'. Let P_i^{CGA} denote the weighted unavailability of function $i \in F$ in the solution of CGA algorithm, and hence $P_{i'}^{\mathrm{CGA}} = Q^{\mathrm{CGA}} = \max_{i \in F} P_i^{\mathrm{CGA}}$.

Since function i' is with the smallest weighted unavailability when we withdraw backup server j', we obtain that $Q^{\mathrm{CGA}} q_{j'} \leq \min_{i \in F} P_i^{\mathrm{CGA}}$. Hence, we obtain,

$$(Q^{\mathrm{CGA}} q_{j'})^{|F|} \leq \prod_{i \in F} P_i^{\mathrm{CGA}} = \prod_{i \in F} w_i p_i \prod_{j \in S} q_j^{c_j}. \tag{5.7}$$

According to Theorem 6, where $Q^* \geq \left(\prod_{i \in F} w_i p_i \prod_{j \in S} q_j^{c_j} \right)^{\frac{1}{|F|}}$, we obtain,

$$Q^{\mathrm{CGA}} \leq \frac{\left(\prod_{i \in F} w_i p_i \prod_{j \in S} q_j^{c_j} \right)^{\frac{1}{|F|}}}{q_{j'}} \leq \frac{Q^*}{q_{j'}}. \tag{5.8}$$

Finally, since $q_{j'} \geq \min_{j \in S} q_j$, we obtain two upper bounds for the CGA algo-

rithm as below,

$$Q^{\mathrm{CGA}} \leq \frac{\left(\prod_{i\in F} w_i p_i \prod_{j\in S} q_j^{c_j}\right)^{\frac{1}{|F|}}}{\min_{j\in S} q_j} \leq \frac{Q^*}{\min_{j\in S} q_j}. \tag{5.9}$$

□

This work provides the upper bound for the CGA algorithm when $w_i p_i$ is the same constant for different functions and $c_j = |F| - 1, j \in S$. This work credits a part of the above proof to the works in [58, 59], where a different problem, which can be viewed to maximize the smallest weighted unavailability while setting $w_i p_i$ to the same constant for different functions and $c_j = 1, j \in S$, was studied.

It becomes difficult to theoretically obtain any upper bound for the CGA algorithm in terms of a general situation, where the values of $w_i, i \in F$, p_i, and $c_j, j \in S$, can be generally set. Addition to the greedy approach, this work introduces another heuristic approach in Section 5.3 to solve the BRAMI problem, where the approximation performance is comprehensively analyzed.

5.3 Linear programming relaxation approach

The LPR approach is a powerful technique to solve such hard optimization problem in (5.4a)-(5.4e) by relaxing the MILP problem to a linear programming (LP) problem, which can be solved in a polynomial time [57]. In this section, it begins by introducing two similar problems of the BRAMI problem presented in Section 5.1. Then, this work develops an LPR approach to solve the BRAMI problem, where a rounding algorithm with polynomial time complexity is introduced to round an optimal solution of the LP problem to a feasible solution of the MILP problem. This work analyzes the approximation performance of introduced rounding algorithm by providing an upper bound for the LPR approach with the introduced rounding algorithm.

5.3.1 Similar problems

In (5.4a)-(5.4e), since $w_i \leq 1$ and $p_i \leq 1$ for function $i \in F$, and $q_i \leq 1$ for backup server $j \in S$, $\log w_i$, $\log p_i$, and $\log q_j$ are with nonpositive values. For

analysis purpose, this work defines $a_i = -\log w_i - \log p_i$, $i \in F$, and $b_j = -\log q_j$, $j \in S$, where $a_i \geq 0$ and $b_j \geq 0$. The minimization problem in (5.4a)-(5.4e) can be transformed to the following equivalent maximization problem.

$$\max \quad r' \tag{5.10a}$$

$$s.t. \quad \text{Eqs. (5.4b), (5.4d), (5.4e)} \tag{5.10b}$$

$$a_i + \sum_{j \in S} x_{ij} b_j \geq r', \forall i \in F, \tag{5.10c}$$

where $r' = -r$. With the form of (5.10a)-(5.10c), the BRAMI problem in this thesis can be viewed as an extension of the generalized load balancing (GLB) problem [57] and the santa claus (SC) problem [62].

For the GLB problem, there is a set of unrelated parallel machines and a set of independent jobs, where each job is associated with a workload. The objective of GLB problem is to assign each job to a machine such that the maximum workload on any machine is minimized. The SC problem is closely related to the GLB problem. The only difference is that the goal of SC problem is to assign all the jobs in a way that maximizes the minimum workload on any machine. The GLB problem, the SC problem, and several variations from them have been extensively studied in computer science and economics [57–63, 110].

Clearly, by considering the functions and backup servers as the machines and jobs, respectively, the BRAMI problem in this thesis is similar to the above two problems. One difference between the BRAMI problem and the GLB problem is that they have opposite objective functions. In addition to that, compared to the GLB and SC problems, there are two main differences in this problem: 1) instead of no job existing on each machine before the assignment, there is an initial workload, which is given by a_i, for function $i \in F$ in this problem; 2) a job is assigned to only one machine in the GLB and SC problems, but a backup server can be assigned to several different functions in this problem.

This work shows how the two points make the BRAMI problem different with the GLB and SC problems. For the first point, in a greedy approach solving the GLB or SC problem, each internal assignment step only depends on the previous assignments of workloads. However, for a similar greedy approach solving the BRAMI problem, such as the SGA or CGA algorithm, each

internal assignment step is affected by both previous assignments of workloads and initial workloads of functions, which indicates that the approximation performance and its analysis of a greedy approach solving the BRAMI problem are different with those of the similar one solving the GLB or SC problem. For the second point, by physically unpacking backup server $j \in F$ as c_j backup servers, each of which has unit capacity, this problem becomes the same as the classical ones in terms of the LPR approach, if the backup server information, such as the failure probability, after unpacking is given. Without the information, we cannot transform this problem to the classical ones.

Based on the ideas introduced in [57, 60] to solve the GLB problem, this work introduces an LPR approach with a rounding algorithm to solve the BRAMI problem. Especially, this work shows how to handle the point that backup server $j \in S$ can be assigned to c_j different functions in the introduced LPR approach.

5.3.2 Linear programming formualtion

In the LPR approach, this work formulates the LP problem of the MILP problem in (5.10a)-(5.10c) by directly setting $x_{ij}, i \in F, j \in S$, to a real decision variable with $0 \leq x_{ij} \leq 1$, which is expressed by,

$$\max \quad r' \tag{5.11a}$$

$$s.t. \quad \text{Eqs. (5.4b), (5.4d), (5.4e), (5.10c),} \tag{5.11b}$$

$$0 \leq x_{ij} \leq 1, \forall i \in F, j \in S. \tag{5.11c}$$

Let L_{MILP} and L_{LP} be the optimal objective values of MILP problem in (5.10a)-(5.10c) and LP problem in (5.11a)-(5.11c), respectively. Clearly, $L_{\text{MILP}} \leq L_{\text{LP}}$, which means that L_{LP} is an upper bound for the optimum of maximization problem in (5.10a)-(5.10c) and $\mathrm{e}^{-L_{\text{LP}}}$ is a lower bound for the worst weighted unavailability in an optimal solution, or $Q^* \geq \mathrm{e}^{-L_{\text{LP}}}$.

In a solution of the MILP problem, the value of $x_{ij}, i \in F, j \in S$ is either 1 or 0; it can be fractional in the range of $[0, 1]$ in a solution of the LP problem. Therefore, given an optimal solution from the LP problem, a rounding algorithm is required to round the optimal solution from the LP problem to a

feasible solution of the MILP problem. More specifically, the rounding algorithm decides the value of $x_{ij}, i \in F, j \in S$ in a fractional solution to either 1 or 0.

5.3.3 Tree-based rounding algorithm

This work develops and analyzes a tree-based rounding (TBR) algorithm based on the concepts of *connected components* [111] and *tree* in graph theory. Given an optimal solution from the LP problem in (5.11a)-(5.11c), it can be represented by an undirected bipartite graph $G(V, E)$, where V and E represent sets of nodes and edges, respectively. In the bipartite graph, functions in F and backup servers in S represent the nodes in two sides, respectively, or $V = F \cup S$; an edge between function $i \in F$ and backup server $j \in S$ exists in E if and only if $x_{ij} > 0$ in the solution.

For function $i \in F$, the total workloads in the solution is the sum of initial workload and assigned workloads, which is expressed by $a_i + \sum_{j \in S} x_{ij} b_j$. L_{LP} is the minimum workload in any function in the solution. For edge $(i, j), i \in F, j \in S$, this work considers the value of $x_{ij} b_j$ as the value of flow, which outgoes from backup server j and incomes to function i. Therefore, the total assigned workloads for a function is the total flow incoming to it.

Given a bipartite graph, this work firstly divides it to several connected components. Some algorithms can be adopt here, such as breadth-first search (BFS) or depth-first search (DFS) [112], where all the connected components are obtained in a linear time proportional to the number of nodes and edges of the graph. Then this work separately considers the assignment in each connected component. Figure 5.2 shows an example connected component, which includes five functions and five backup servers. In Fig. 5.2, the number attached to edge (i, j) indicates the flow value $x_{ij} b_j$ on the edge.

In a connected component, there may exist some cycles, such as $(i_1, j_1, i_2, j_2, i_1)$ in Fig. 5.2. In the TBR algorithm, this work modifies the given solution by eliminating all the cycles such that each modified connected component can be represented by a tree; the final rounding is performed based on the tree structure. This work firstly introduces how to eliminate cycles and perform rounding in a special case, where a backup server can be assigned to only one

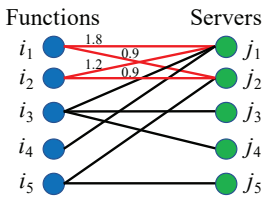


Figure 5.2: Connected component with cycle.

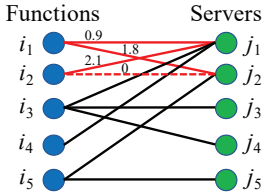


Figure 5.3: Modified connected component with no cycle.

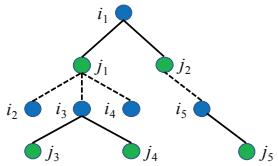


Figure 5.4: Integral assignment.

function (see Algorithm 9), or $c_j = 1, \forall j \in S$. Then this work extends the TBR algorithm to a general case (see Algorithm 10), where $c_j \geq 1, \forall j \in S$, with keeping the same computational time complexity and approximation performance with those analyzed in the special case.

Special case of $c_j = 1$

When $c_j, j \in F$, is set to 1 in (5.11b), this work considers to eliminate all the cycles in any connected component with the following procedure, which is defined as *elimination procedure 1*. In a bipartite graph, any cycle has even number of edges. Given a cycle with k edges, where k is an even number in the range of $[4, \min(2|F|, 2|S|)]$, this work numbers each edge along the cycle starting from an edge with minimum flow δ. The cycle can be represented by $(e_1, e_2, \cdots, e_k)$, where e_l, $l \in [1, k]$, denotes an edge with number l in the cycle. To eliminate the cycle, this work decreases the flow in all edges with

Algorithm 9: Tree-based rounding algorithm in special case of $c_j = 1$

Input: A fractional solution, $G(V, E)$, from the LP problem
Ouput: An integral solution for the MILP problem
Partition $G(V, E)$ into a set of connected components
for each connected component **do**
  Eliminate all the cycles by using elimination procedure 1
  for each backup server j **do**
    Assign backup server j to its parent function
  end
end

odd numbers and increase the flow in all edges with even numbers by the same amount δ. Then all edges with zero flow are deleted from the solution. For remaining edge $(i, j), i \in F, j \in S$, let x'_{ij} represent x_{ij} after the modification, hence $x'_{ij}b_j$ is the modified flow on edge (i, j). Note that, with the above procedure, this work can eliminate all the cycles in the given solution without adding any new edge or changing the total incoming (outgoing) flow for each function (backup server). In other words, the modified solution still satisfies all the constraints in (5.11a)-(5.11c) and keeps the objective value as L_{LP}. For example, by considering edge (i_2, j_2) as the first edge and $\delta = 0.9$ in the cycle of Fig. 5.2, the connected component in Fig. 5.2, where $b_{j_1} = 4$ and $b_{j_2} = 3$ in the special case, can be modified to the one in Fig. 5.3 with deleting edge (i_2, j_2).

After eliminating all the cycles, each connected component can be represented by a tree, where an arbitrary function is selected as the root. For each node, there may be a parent and some children nodes adjacent with it. For example, on the tree in Fig. 5.4, the parent and children backup servers of function i_3 are $\{j_1\}$ and $\{j_3, j_4\}$, respectively; the parent and children functions of backup server j_1 are $\{i_1\}$ and $\{i_2, i_3, i_4\}$, respectively. Let V_i and W_i be the sets of parent and children of function i on the tree, respectively. Note that each backup server has exactly one parent; each function has at most one parent, or the size of V_i does not exceed one. Finally, this work rounds the given fractional solution to an integral solution by assigning each backup server only to its parent function, or setting $x^*_{ij} = 1, \forall i \in F, j \in W_i$ and $x^*_{ij} = 0, \forall i \in F, j \in V_i$,

where x^*_{ij} represents the assignment between backup server j and function i in the rounded integral solution. For example, an integral assignment of the fractional assignment in Fig. 5.3 can be expressed in Fig. 5.4, where a solid line and a dotted line represent rounding the corresponding fractional x'_{ij} to 1 and 0, respectively.

This work analyzes the computational time complexity of TBR algorithm in the special case of $c_j = 1, j \in S$. This work assumes that $|S||F| \geq |S| + |F|$[1] to simplify $O(|S||F| + |S| + |F|)$ as $O(|S||F|)$. Initially, there are at most $|S||F|$ edges in $G(V, E)$. The computational time complexity of dividing $G(V, E)$ into several connected components is $O(|S||F|)$ by adopting BFS or DFS. For a given connected component, we can use BFS or DFS to find a cycle, or confirm if there is any cycle, in $O(|S||F|)$ time. Hence, before each elimination procedure, we can specialize a cycle in $O(|S||F|)$ time. In each elimination procedure, since the maximum number of edges in a cycle is $\min(2|F|, 2|S|)$, it takes $O(|S|)$ time to determine δ, where $|F| \geq |S|$ is considered. Decreasing flow with δ for the edges with odd numbers, increasing flow with δ for the edges with even numbers, and deleting the edges with zero flow, require $O(|S|)$ time. Totally, a cycle can be found and eliminated in $O(|S||F|)$ time. Since we delete at least one edge in each elimination procedure, we can run the elimination procedure $O(|S||F|)$ times. As a result, we can eliminate all the cycles in $O(|S|^2|F|^2)$ time. Since we always start from an arbitrary function node when we run BFS or DFS to detect a cycle, a tree with function node root can be parallel set up for each connected component after all the elimination procedures. Finally, we take $O(|S|)$ time to assign each backup server to its parent function. Therefore, the computational time complexity of TBR in the special case is $O(|S|^2|F|^2)$.

This work analyzes the approximation performance of above algorithm. Let Q^{TBR} represent the worst weighted unavailability obtained by the LPR approach with the TBR (LPR-TBR) algorithm.

Theorem 8 *In the special case, by using Algorithm 9, we can obtain a feasible assignment of backup servers to functions with weighted unavailability of function at most* $\mathrm{e}^{-(L_{\text{LP}} - \max_{j \in F} b_j)}$*, or* $Q^{\text{TBR}} \leq \mathrm{e}^{-(L_{\text{LP}} - \max_{j \in F} b_j)}$.

Proof : Consider an arbitrary function $i \in F$. Since elimination procedure

[1]This condition is satisfied in the case of $|S| \geq 2$ and $|F| \geq 2$.

1 does not change the total incoming flow in any function, we obtain,

$$a_i + \sum_{j \in V_i \cup W_i} b_j \geq a_i + \sum_{j \in V_i \cup W_i} x'_{ij} b_j \geq L_{\mathrm{LP}}. \tag{5.12}$$

For the parent backup server $j \in V_i$ of function i, $b_j \leq \max_{j' \in F} b_{j'}$. Therefore, we obtain,

$$a_i + \sum_{j \in W_i} b_j \geq L_{\mathrm{LP}} - \max_{j \in F} b_j. \tag{5.13}$$

Hence, by using Algorithm 9, the rounded solution for (5.10a)-(5.10c) is lower bounded by $(L_{\mathrm{LP}} - \max_{j \in F} b_j)$ and the obtained worst weighted unavailability of function is upper bounded by $\mathrm{e}^{-(L_{\mathrm{LP}} - \max_{j \in F} b_j)}$, or $Q^{\mathrm{TBR}} \leq \mathrm{e}^{-(L_{\mathrm{LP}} - \max_{j \in F} b_j)}$. □

General case of $c_j \geq 1$

For the general case, where each backup server can be assigned to several different functions, we cannot directly use the above procedure to eliminate all the cycles in a given fractional solution. The main reason is that elimination procedure 1 cannot guarantee to always satisfy constraint (5.11c) in the general case.

This work also takes cycle $(i_1, j_1, i_2, j_2, i_1)$ in Fig. 5.2 as an example. Now consider $b_{j_1} = 2, b_{j_2} = 1.5, c_{j_1} = 2$, and $c_{j_2} = 2$ in the general case. Based on flow value $x_{ij} b_j$ indicated on the corresponding edge (i, j) in the cycle, we consider $x_{i_1 j_1} = 0.9$, $x_{i_1 j_2} = 0.6$, $x_{i_2 j_1} = 0.6$, and $x_{i_2 j_2} = 0.6$. From the result after elimination procedure 1, which is shown in Fig. 5.3, we observe that the flow in edges (i_1, j_2) and (i_2, j_1) exceed b_{j_2} and b_{j_1}, respectively. In other words, the values of x'_{ij} in edges (i_1, j_2) and (i_2, j_1) are greater than 1, which violate constraint (5.11c). In a modified solution, if constraint (5.11c) is not satisfied for backup server $j \in S$, or $\exists i \in F$ such that $x'_{ij} > 1$, the number of edges including backup server j may less than c_j. As a result, we may not be able to round the modified fractional solution to a feasible integral solution.

To address this issue, this work extends elimination procedure 1 to *elimination procedure 2* as follows. Similar with elimination procedure 1, this work numbers each edge along the given cycle starting from an edge with minimum

Algorithm 10: Tree-based rounding algorithm in general case of $c_j \geq 1$

Input: A fractional solution, $G(V, E)$, from the LP problem
Ouput: An integral solution for the MILP problem
Partition $G(V, E)$ into a set of connected components
for each connected component **do**
  Eliminate all the cycles by using elimination procedure 2
  for each backup server j **do**
    Assign backup server j to its parent function
    if $c'_j > 1$ **then**
      Assign backup server j to its $c'_j - 1$ child functions with the greatest weighted unavailabilities
    end
  end
end

flow δ. For edge $e_l = (i, j)$, where l is an even number, this work computes the remaining flow admissible for edge e_l, which is defined by $\gamma_{e_l} = (1 - x_{ij})b_j$; the minimum remaining flow on any edge with even number is represented by γ. If $\delta \leq \gamma$, this work just eliminates the cycle by using elimination procedure 1. If $\delta > \gamma$, in order to avoid the violation of constraint (5.11c), this work decreases the flow in all edges with odd numbers and increase the flow in all edges with even numbers by the same amount γ, instead of δ. Let H denote the set of edges, where $x'_{ij} = 1$ after adding γ amount flow. For edge $(i, j) \in H$, this work directly pre-assigns backup server j to function i before the final rounding procedure. For backup server $j \in S$, this work decreases its remaining capacity by one after each pre-assignment. Then, in the modified connected component, this work deletes all the edges in H to eliminate the cycle, and do not change the modified flow in other edges. Let c''_j represent the remaining capacity of backup server $j \in S$ after eliminating all the cycles; it can be proved that $c''_j > 0$ is always satisfied. Let D_i represent a set of backup servers, each of which is pre-assigned to function i in the elimination procedure. Note that, with elimination procedure 2, we can eliminate all the cycles in a given solution for the general case; the modified solution, including the

pre-assignments, still satisfies all the constraints in (5.11a)-(5.11c) and keeps the objective value as L_{LP}. For example, for cycle $(i_1, j_1, i_2, j_2, i_1)$ in Fig. 5.2, $\delta = 0.9$ and $\gamma = 0.6$. After modifying the flow in each edge, this work directly pre-assigns backup servers j_2 to function i_1, and delete edge (i_1, j_2) in the given connected component to eliminate the cycle. Figure 5.5 shows the modified connected component of that in Fig. 5.2 and the pre-assignment in this general case.

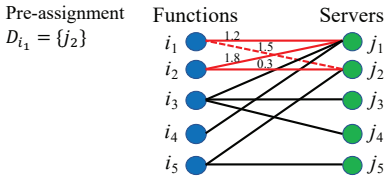


Figure 5.5: Modified solution with no cycle in this general case.

After eliminating all the cycles by elimination procedure 2, each connected component can be represented by a tree, where an arbitrary function is selected as the root. For backup server $j \in F$, this work firstly assigns it to its parent function; if $c''_j > 1$, this work then assigns it to its $c''_j - 1$ child functions with the greatest weighted unavailabilities. In other words, x^*_{ij} is set to 1, $\forall i \in F, j \in W_i$ and x^*_{ij} is probably set to either 1 or 0, $\forall i \in F, j \in V_i$. For example, a final integral assignment of the fractional assignment in Fig. 5.5 can be expressed in Fig. 5.6, where a solid line and a dotted line represent rounding the corresponding fractional x'_{ij} to 1 and 0, respectively.

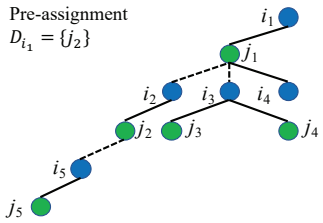


Figure 5.6: Integral assignment in this general case.

In elimination procedure 2, compared to elimination procedure 1, this work only introduces the additional computation to obtain γ, whose computational

time complexity is $O(|S|)$. Hence, this work still takes $O(|S|^2|F|^2)$ time to eliminate all the cycles. In the final rounding procedure, sorting all the child functions and completing the assignment need $O(|F|\log|F|)$ time for each backup server, and totally $O(|S||F|\log|F|)$ time is required. Therefore, the computational time complexity of TBR algorithm in the general case is still $O(|S|^2|F|^2)$.

For the approximation performance of TBR algorithm in the general case, this work obtains the following theorem.

Theorem 9 *In the general case, by using Algorithm 10, we can obtain a feasible assignment of backup servers to functions with weighted unavailability of function at most* $\mathrm{e}^{-(L_{\mathrm{LP}}-\max_{j\in F} b_j)}$*, or* $Q^{\mathrm{TBR}} \leq \mathrm{e}^{-(L_{\mathrm{LP}}-\max_{j\in F} b_j)}$.

Proof : Consider an arbitrary function $i \in F$. Since elimination procedure 2 does not change the total incoming flow in any function, we obtain,

$$a_i + \sum_{j\in D_i} b_j + \sum_{j\in V_i\cup W_i} b_j \geq a_i + \sum_{j\in D_i} b_j + \sum_{j\in V_i\cup W_i} x'_{ij} b_j \geq L_{\mathrm{LP}}. \tag{5.14}$$

The parent backup server $j \in V_i$ of function i is probably assigned to function i, where $b_j \leq \max_{j'\in F} b_{j'}$. Therefore, we obtain,

$$a_i + \sum_{i\in D_i} b_j + \sum_{j\in W_i} b_j + \sum_{j\in V_i} x^*_{ij} b_j \geq L_{\mathrm{LP}} - \max_{j\in F} b_j. \tag{5.15}$$

Hence, in the general case, by using Algorithm 10, we obtain the same approximation performance with that in the special case; the worst weighted unavailability of function in the rounded solution is upper bounded by $\mathrm{e}^{-(L_{\mathrm{LP}}-\max_{j\in F} b_j)}$, or $Q^{\mathrm{TBR}} \leq \mathrm{e}^{-(L_{\mathrm{LP}}-\max_{j\in F} b_j)}$. □

This implies that, when $\max_{j\in F} b_j$ is fixed, the greater L_{LP} is, the better approximation the LPR-TBR algorithm can provide.

5.4 Numerical results

In the numerical analysis, this work compares the introduced approaches with a scheme which ignores the importance of functions and backup server failures in terms of the exact worst weighted unavailability in Section 5.4.2. Then this work compares different approaches introduced in this thesis in Sections 5.4.3

and 5.4.4. This work considers the BRAMI problem with small size, such as the number of functions and backup servers is small, and the MILP problem in (5.4a)-(5.4e) can be solved in an admissible time to obtain the optimal solution. When the size of BRAMI problem becomes large, this work considers the best feasible solution obtained from an optimization solver within the admissible computation time. This work evaluates the heuristic approaches introduced in Sections 5.2 and 5.3 by comparing their performances and approximation guarantees to the optimal solution in Section 5.4.3 and to the best feasible solution in Section 5.4.4.

This work uses Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory to solve the MILP problem or run the heuristic algorithms. The MILP problem is solved by the IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer with version 12.8 [88].

5.4.1 Experiment setup

The work in [22] describes a mean number of 3.5 failures per year in the worst case among four types of middleboxes where the 95th and 99th percentiles for downtime are about 2.5 and 17.5 days, respectively. In literature, the failure probability of a network function is always considered as the ratio of its downtime to the total time of uptime and downtime [51, 52]. Therefore, this work considers that the failure probability of each function is uniformly distributed over the range of [0.025, 0.175] where $0.025 \approx \frac{3.5\times2.5}{365}$ and $0.175 \approx \frac{3.5\times17.5}{365}$, which is the same estimation with that in [19]. Since, compared to middleboxes, backup servers may process workloads less frequently and be less depreciated, this work considers that the failure probability of a backup server can be lower than that of a middlebox, and it is set to be uniformly distributed over the range of [0.01, 0.05] in the experiment; the performances with different settings of the range of q_j are compared in Fig. 5.8.

Since the performances of heuristic approaches depend on the actual input, this work adopts average case analysis in the numerical analysis. This work conducts 500 trials, in each of which, unless specifically stated, the failure probability of each function, the weight of each function, and the failure probability of each backup server are uniformly distributed over the ranges of

[0.025, 0.175], [0.01, 1], and [0.01, 0.05], respectively.

5.4.2 Comparison with scheme that ignores importance of functions and backup server failures

This work considers the balanced assignment (BA) approach presented in [19] as the scheme which ignores the importance of functions and backup server failures. To compare introduced approaches with the BA approach, this work considers the BRAMI problem with a special case that the information of $c_j = c$ functions is synchronized to backup server $j \in S$ and only one of them can be recovered at the same time due to the limited computing resources, or $c_j = c \geq c'_j = 1$, $\forall j \in S$.

In the experiment, this work sets $|F| = 10$ and c is uniformly distributed in the range of [1, 5] over trials. Given a BRAMI problem with the special case, introduced approaches and the BA approaches solve it with assuming $c_j = c'_j = c$ and with ignoring the importance of functions and backup server failures, respectively. This work computes the exact worst weighted unavailability for a given solution.

Figure 5.7 shows the average values of exact worst weighted unavailabilities which are obtained by introduced approaches and the BA approach for different numbers of backup servers $|S|$. This work observes that the introduced approaches outperform the BA approach in terms of the exact worst weighted unavailability; especially, the exact worst weighted unavailability obtained by the MILP approach, which considers the importance of functions and backup server failures, is about seven times smaller than that obtained by the BA approach.

5.4.3 Competitive evaluation for small size problem

In Sections 5.4.3 and 5.4.4, this work considers a large-size network with 100 middleboxes [113], or $|F| = 100$, and the capacity of each backup server is set to be uniformly distributed over the range of [1, 15].

Figure 5.8(a) shows the comparison among the average values of worst weighted unavailabilities, which are obtained by running the heuristic algo-

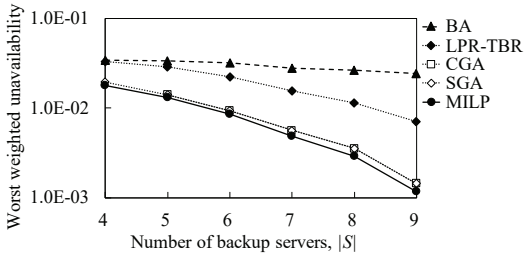
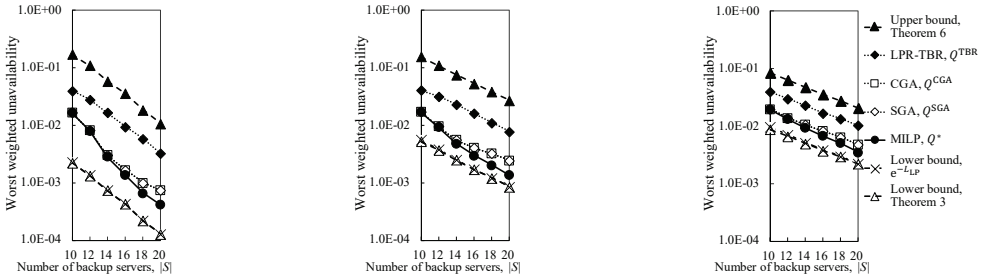


Figure 5.7: Comparison between introduced approaches and BA approach.

rithms and by solving the MILP problem, and the lower and upper bounds, for different numbers of backup servers $|S|$, when the range of q_j is set to [0.01, 0.05]. This work observes that the worst weighted unavailabilities obtained by different approaches and the lower and upper bounds decrease as the number of backup servers increases. It is obvious that the lower bound in Theorem 6 is a decreasing function in terms of the number of backup servers. For the obtained worst weighted unavailabilities and the lower bound of $e^{-L_{LP}}$, with more backup servers, more protections are provided to the function that is in the worst case. As a result, the worst weighted unavailability and the value of $e^{-L_{LP}}$ decrease. For the upper bound in Theorem 9, since the lower bound of failure probability of backup sever is fixed, which is approximate to fixing the value of $\max_{j \in F} b_j$, the upper bound of $e^{-(L_{LP} - \max_{j \in F} b_j)}$ decreases as the value of $e^{-L_{LP}}$ decreases.



(a) $[0.01, 0.05]$ (b) $[0.025, 0.175]$ (c) $[0.1, 0.25]$

Figure 5.8: Comparison among average values of worst weighted unavailabilities obtained by different approaches and lower and upper bounds; different ranges of q_j are considered in different subfigures; legend in Fig. 5.8(c) is applied to each subfigure.

For the two lower bounds, this work observes that they are comparable and the lower bound of $e^{-L_{LP}}$ is slightly tighter than the one in Theorem 6. This is because that the lower bound in Theorem 6 corresponds to a fractional assignment, which is the most balanced assignment where the result weighted unavailabilities of all functions are the same. However, this balanced assignment may not be a feasible fractional solution of the LP version of BRAMI problem for two reasons. Firstly, it is because that the value of $w_i p_i$ of function $i \in F$ may be assigned to other functions for balancing purpose, which is not allowed in the BRAMI problem. Secondly, the most balanced assignment may not satisfy constraint (5.4d). Therefore, $e^{-L_{LP}}$, which is the worst weighted unavailability in the optimal solution of LP version of BRAMI problem, is greater than or equal to the lower bound in Theorem 6.

The deviation between a heuristic result and the optimal value is defined by the value of heuristic result divided by the optimal value, where $\gamma_{SGA} = \frac{Q^{SGA}}{Q^*}$ represents the deviation between the result from the SGA algorithm and the optimal value. The average computation times of running the heuristic algorithms and solving the MILP problem are represented by T_{SGA}, T_{CGA}, T_{TBR}, and T_{MILP}, respectively.

Tables 5.1 and 5.2 show the average computation times of different approaches and the deviations corresponding the results shown in Fig. 5.8(a), respectively. This work observes that T_{MILP} increases as the number of backup servers increases. This work observes that the average computation times of running the SGA algorithm, the CGA algorithm, and the LPR-TBR algorithm to obtain solutions are about 10^5, 10^4, and 10^2 times less than that of solving the MILP problem, respectively; the SGA algorithm is the fastest one among all the approaches in the examined scenarios.

Table 5.1: Average computation time (seconds) of different approaches.

$\|S\|$	T_{MILP}	T_{SGA}	T_{CGA}	T_{TBR}
10	14.66	8.19×10^{-4}	1.60×10^{-3}	1.25×10^{-1}
12	28.21	8.29×10^{-4}	1.60×10^{-3}	1.25×10^{-1}
14	42.38	7.44×10^{-4}	1.27×10^{-3}	1.15×10^{-1}
16	53.73	8.45×10^{-4}	1.31×10^{-3}	1.30×10^{-1}
18	57.79	8.01×10^{-4}	1.49×10^{-3}	1.37×10^{-1}
20	84.31	2.00×10^{-3}	4.23×10^{-3}	3.37×10^{-1}

Table 5.2: Deviations from optimal values.

$\|S\|$	γ_{SGA}	γ_{CGA}	γ_{TBR}
10	1.00	1.00	2.40
12	1.01	1.01	3.49
14	1.06	1.06	5.83
16	1.22	1.22	7.16
18	1.53	1.53	8.81
20	1.80	1.80	7.81

From Fig. 5.8(a) and Tables 5.1 and 5.2, this work observes that the results from the SGA algorithm and the CGA algorithm are comparable and closer to the optimal values than those from the LPR-TBR algorithm. There are some pros and cons for the three heuristic algorithms. On one hand, the CGA algorithm and the SGA algorithm may provide less deviations and require shorter computation time than the LPR-TBR algorithm does. However, there is no approximation guarantee for these two algorithms. On the other hand, the worst weighted unavailability is upper bounded by adopting the LPR-TBR algorithm, which may perform worse than the CGA algorithm and the SGA algorithm in terms of deviation and computation time. With the analyses of above pros and cons, a network operator can decide which heuristic approach should be adopted according to the requirements in specific application scenarios.

Figures 5.8(b) and 5.8(c) show the results with different settings of the range of q_j compared to Fig. 5.8(a). this work observes that as the average value of q_j increases, the result of each approach or bound increases except for that obtained by the upper bound in Theorem 9. The value of $\mathrm{e}^{-L_{\text{LP}}}$ increases as the average value of q_j increases, but the value of $\mathrm{e}^{\max_{j\in F} b_j}$ decreases when the lower bound of q_j increases. As a result, the value of $\mathrm{e}^{-(L_{\text{LP}}-\max_{j\in F} b_j)}$ does not always increase. For the results of different approaches and bounds, we have the similar observations in both Figs. 5.8(b) and 5.8(c) compared to those in Fig. 5.8(a).

5.4.4 Competitive evaluation for large size problem

Let T denote the admissible computation time (seconds) in the experiment. When the size of BRAMI problem becomes large, the optimal value cannot

be obtained within T, such as $|S| \geq 20$ when T is considered as 60. In such cases, this work considers the best feasible solution obtained by solving the MILP problem within T, where T is set large enough for running any heuristic algorithm.

Figure 5.9 shows the comparison among the average values of worst weighted unavailabilities, which are obtained by running the heuristic algorithms and by solving the MILP problem within different values of T, and the lower and upper bounds, for different numbers of backup servers $|S|$. For $T = 5$, this work observes that the worst weighted unavailability in the best feasible solution obtained by the MILP approach is greater than those from the SGA algorithm and the CGA algorithm for any value of $|S|$; compared with the LPR-TBR algorithm, the MILP approach outperforms it when $|S| \leq 45$ and preforms worse than it when $|S| > 45$ due to the increase of problem size. For $T = 15$, this work observes that the MILP approach always outperforms the LPR-TBR algorithm; the MILP approach preforms better than the SGA algorithm and the CGA algorithm when $|S| \leq 35$, and it performs worse when $|S| > 35$. For $T = 60$, this work observes that, for different values of $|S|$, the worst weighted unavailabilities in the best feasible solutions by solving the MILP problem are always the least among all the results by different approaches.

Figure 5.10 shows the dependency of the average value of worst weighted unavailabilities obtained by the MILP approach on the admissible computation time T when $|S| = 50$. This work observes that at the beginning of $T = 5$, the MILP approach provides the greatest worst weighted unavailability compared to other results obtained by the heuristic approaches. As the value of T increases, the worst weighted unavailabilities in the best feasible solution obtained by solving the MILP problem decreases. This is because that the longer computation time can provide the result nearer to the optimal one. This work observes that the worst weighted unavailability obtained by the MILP approach decreases much faster as T increases when $T \leq 25$ than it does when $T > 25$. It indicates that the MILP approach may need a long computation time to obtain the optimal solution, but a relatively much shorter time to obtain a solution comparable to the optimal one. When T is set to greater than about 20, this work observes that the worst weighted unavailabilities in the best feasible solutions obtained by solving the MILP problem are

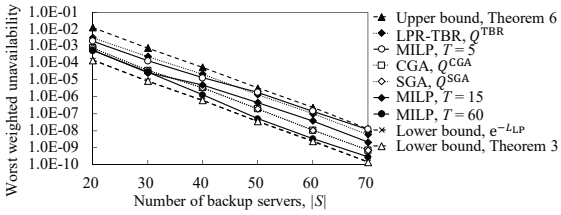


Figure 5.9: Comparison among average values of worst weighted unavailabilities obtained by different approaches within different values of T and lower and upper bounds.

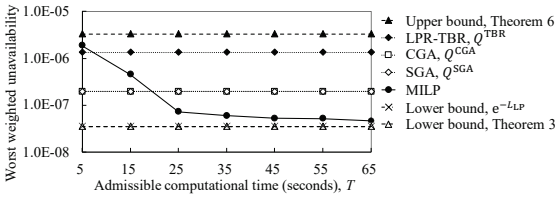


Figure 5.10: Dependency of average value of worst weighted unavailabilities obtained by MILP approach on admissible computation time T when $|S| = 50$.

always the least among all the results by different approaches, which means that solving the MILP problem is the best approach in terms of the objective value when $T > 20$ in the examined scenarios. Furthermore, by referring to the lower bound of $e^{-L_{LP}}$, this work obtains that the deviation between the objective value in the best feasible solution obtained by the MILP approach and the optimal value is at most 1.31 when $T = 65$.

In summary, the admissible computation time T is another significant factor for a network operator to choose an appropriate approach. For some delay-sensitive application scenarios, the heuristic approaches should be adopted to provide an assignment within a short time, such as the SGA algorithm can terminate within 10^{-2} seconds for $|F| = 100$ and $|S| = 50$ in the experiment. For other scenarios, where the requirement on T can be relaxed to a certain degree, the best feasible solution by solving the MILP problem within T may provide a good approximation to the optimum.

5.5 Chapter summary

This chapter proposed a backup resource allocation model for middleboxes with considering both failure probabilities of network functions and backup servers. This work took the importance of functions into account by defining a weighted unavailability for each function. This work aimed to find an assignment of backup servers to functions where the worst weighted unavailability is minimized. This work formulated the BRAMI problem as an MILP problem. This work proved that the BRAMI problem is NP-complete. Two heuristics based on the greedy approach and one based on the LPR approach were introduced to solve the same optimization problem. The computational time complexities of three heuristic algorithms were analyzed as polynomials. This work analyzed the approximation performances of different heuristic algorithms by providing several lower and upper bounds. This work compared among the results obtained by different approaches and the lower and upper bounds. The results showed the pros and cons of different approaches. When the BRAMI problem becomes large, solving the MILP problem needs a long computation time to obtain the optimal solution, but a relatively much shorter time to obtain a solution comparable to the optimal one. Heuristic approaches outperform the MILP approach when the admissible computation time is set short. The CGA algorithm and the SGA algorithm provide less deviations and require shorter computation time than the LPR-TBR algorithm does. However, only the performance of LPR-TBR algorithm has an upper bound and there is no approximation guarantee for the other two heuristics. Referring to the analyses, a network operator can choose an appropriate approach according to the requirements in specific application scenarios.

Chapter 6

Unavailability-aware shared backup allocation model

This thesis proposes an unavailability-aware backup allocation model with the shared protection for middleboxes with comprehensively considering heterogeneous procedures of functions and backup servers. A part of the work in this chapter was presented in [50]. In this model, multiple functions can share the backup resources on backup servers. The proposed model aims to find the assignment of backup servers to functions to minimize the maximum unavailability among functions. This work develops an analytical approach based on the queueing theory to compute the unavailability of middleboxes for a given backup allocation. The heterogeneous failure, repair, recovery, and waiting procedures of functions and backup servers, which lead to several different states for each function and for the whole system, are considered. In the analytical approach, this work analyzes what all system states are, how they transit from/to each other, and what the equilibrium-state probability of each system state is. The unavailability of each function is estimated based on the obtained probabilities of system states. This work introduces an SA heuristic to solve the backup allocation problem by using the developed analytical approach. The performance dependencies on the failure, repair, and recovery behaviors of functions and backup servers are evaluated. This work compares the proposed model with a baseline model. The results reveal that the proposed unavailability-aware model reduces the maximum unavailability 16% in

average compared to the baseline model in the examined scenarios.

The rest of this chapter is organized as follows. Section 6.1 presents the proposed model. Section 6.2 describes the analytical approach based on the queueing theory. Section 6.3 introduces the heuristic algorithm to solve the backup allocation problem. The performances are evaluated in Section 6.4. Section 6.5 summaries this chapter.

6.1 Model and problem definition

Let F and S denote a set of functions and a set of backup servers in a network, respectively. In this thesis, this work assumes that sufficient and reliable networking resources are provided. This work studies the assignment between backup servers and functions with considering the limited computing resources and the heterogeneous procedures that can occur at a network node, function or backup server, with affecting the unavailability of functions.

6.1.1 Shared protection for functions

To protect a function against failure events, a backup server is assigned to protect it. This work assumes that, for different functions, the amounts of resources utilized for information synchronization or recovery on the backup servers are the same [19, 24, 48, 49]. Backup server $j \in S$ can be assigned to protect at most c_j functions whose information required for recovery is updated to backup server j, where at most r_j functions can be recovered at the same time and $r_j \leq c_j$. Let x_{ij} denote a binary variable; if function $i \in F$ is protected by backup server $j \in S$, $x_{ij} = 1$; otherwise, $x_{ij} = 0$.

Figure 6.1 shows an example of backup allocation for three functions protected by two backup servers, where a link exists between function $i \in F$ and backup server $j \in S$ if $x_{ij} = 1$. Backup server j_1 can protect two functions and only one of them can be recovered at the same time. It indicates that functions i_1 and i_2 share the backup resource on backup server j_1; if functions i_1 and i_2 fail simultaneously, only one of them can be recovered at the same time.

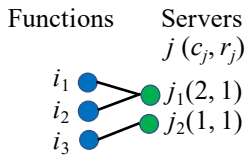


Figure 6.1: Example of backup allocation with shared protection.

6.1.2 Heterogeneous procedures

In the network, each element in $W = F \cup S$ may experience different procedures that affect the unavailability of functions; for the same procedure, the behaviors may vary for different elements. This work describes these procedures.

Failure procedure

This work considers that an element in W is either in an active state or failed state. This work assumes that each element, function or backup server, in W experiences failure events independently to other elements, where the failure events occur at an element based on a Poisson process when the element is in the active state. For each element, the average failure rate of corresponding Poisson process can be estimated by sampling the status of element within a certain period.

In literature, several studies introduced that the average failure rate of an element depends on its age or running time, which can be a key feature to classify elements in W to several classes, in each of which the elements have the same average failure rate [114, 115]. Let $H = [1, |H|]$ denote the set of classes for the elements in W; an element in class $h \in H$ has the average failure rate of λ_h per unite time.

Repair procedure

Once a failure event occurs to an element in W, it goes to the failed state; a procedure is started to repair the element itself to return to the active state. For example, if a function or backup server fails due to a broken hardware component on its hosting physical machine, some methods, such as replacing the broken component, are taken to repair the physical machine, which makes

the failed function or backup server return to the active state. This work considers that the time to complete the repair procedure of a failed function or backup server in class $h \in H$ follows an exponential distribution with the average value of μ_h^{-1} unit time.

Recovery procedure

When a function in F is in the failed state, it needs to be recovered on a backup server before the completion of its repair procedure[1]. Once the repair procedure is completed, the failed function goes to the active state and the corresponding recovered one if any on the backup server is released. In other words, the function recovered on a backup server is switched back to a function in the active state. When a backup server in S is in the active state, it can recover the failed functions if the recoveries do not exceed its capacity. When a backup server in S is in the failed state, it cannot recover any function.

The operations to recover a failed function, such as launching a virtual machine and recreating the missing states, on an active backup server take time. This work assumes that the recovery time for a failed function on backup server $j \in S$ follows an exponential distribution with the average value of δ_j^{-1} unit time.

Waiting procedure

For a function, there are two cases where the protection prepared for it fails such that the recovery procedure cannot start immediately. One case is that a function and its backup server are in the failed state at the same time. For example, when both function i_3 and backup server j_2 in Fig. 6.1 are in the failed state, the prepared protection for function i_3 fails. The other case is that a function is in the failed state and its preassigned backup server is in the active state, but no capacity remains on the backup server to recover the failed function. For example, when function i_2 in Fig. 6.1 fails, if backup server j_1 is hosting failed function i_1, the prepared protection for function i_2 fails.

[1]In this thesis, for a failed function, the terminologies of *recover* and *repair* refer to recovering it on a backup server and making itself return to the active state through the repair procedure, respectively.

To recover a failed function when its protection fails, different policies can be developed based on different application scenarios. In this thesis, this work considers a preassigned backup only policy (PBOP), where the failed function can only be recovered by its preassigned backup server. If the prepared protection fails, the failed function needs to wait. PBOP may be adopted in some scenarios with limited network bandwidth resources to avoid the additional transmissions for information required to recover failed functions [23].

6.1.3 State transition and unavailable time for each function

Figure 6.2 shows the state transition for each function with considering all procedures. When an active function goes to the failed state, if its protection does not fail, the recovery procedure on the preassigned backup server starts; otherwise, the function waits to start the recovery on its backup server. When the preassigned backup server is active and has remaining capacity for recovery, the recovery procedure of failed function starts. In the case that multiple failed functions protected by the same backup server are in the waiting procedure, this work considers randomly choosing some of them to start the recoveries which do not exceed the maximum capacity of backup server. After the completion of recovery procedure, the failed function is recovered on the backup server and becomes available. For a failed function that is recovered or is being recovered on a backup server, if the backup server fails, the function goes to the waiting procedure. When the repair procedure of a failed function is completed, the failed function goes to the active state.

The function is considered as unavailable during the waiting and recovery procedures. The time switching the function recovered on a backup server to a function that goes to the active state is assumed much shorter than the recovery time. In other words, the unavailable time of a function consists of two types of time, which are waiting time and recovery time.

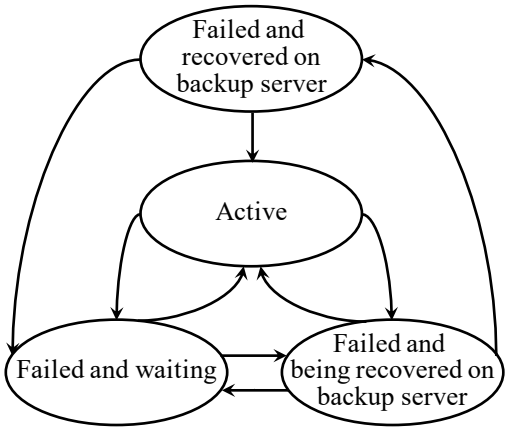


Figure 6.2: State transition for each function.

6.1.4 Unavailability of function

In literature, such as the works in [51, 52], the unavailability of function $i \in F$ is often considered as the ratio of its average unavailable time to the sum of its average available time and average unavailable time, which can be define as,

$$Q_i = \frac{T_i}{T_i + T_i'}, \tag{6.1}$$

where T_i and T_i' represent the average unavailable time and the average available time of function i, respectively. Clearly, if there is no backup that is used to recover function i with class $h \in H$, its unavailability is $Q_i = \frac{\mu_h^{-1}}{\mu_h^{-1}+\lambda_h^{-1}}$.

6.1.5 Problem definition

The unavailability of a function depends on the backup allocation for all functions. Let $J = \max_{i \in F} Q_i$ denote the maximum unavailability among functions. The unavailability-aware shared backup allocation (UASBA) problem is defined as follows:

Problem G iven sets of functions and backup servers, the average failure rate and average repair time for each function and backup server, and the average recovery time on a backup server, find the optimal backup allocation for all functions to minimize the maximum unavailability among functions, or J.

To address the UASBA problem, we first need to know how the unavailability of a function is for a given solution of backup allocation. This work answers this question in Section 6.2 by developing an analytical approach based on the queueing theory.

6.2 Analyses for unavailability based on queueing theory

Let $L_j = \{i | i \in F : x_{ij} = 1\}$ denote the set of functions protected by backup server $j \in S$; $L_j^h \subseteq L_j$ represents the set of all functions with class $h \in H$ in L_j. This work considers backup server $j \in S$ and L_j as a group. Since a failed function can only be recovered by its preassigned backup server in PBOP, this work analyzes the unavailabilities of functions in a group independently of other groups. Two functions with the same classes in the same group have the same failure, repair, and recovery behaviors, which leads to the same unavailability for them.

This work considers a state of functions of class $h \in H$ in the group containing backup server $j \in S$ as (m_h, n_h, o_h, p_h), where m_h, n_h, o_h, and p_h represent the numbers of functions of class h which are active, waiting, being recovered, and recovered, respectively. Clearly, the values of m_h, n_h, o_h, and p_h satisfy $\sum_{h \in H}(m_h + n_h + o_h + p_h) = |L_j| \leq c_j$ and $\sum_{h \in H}(o_h + p_h) \leq r_j$. A state of backup server j is considered as (q), where $q \in \{0, 1\}$; $q = 1$ and 0 mean that backup server j is in the active and failed states, respectively. A system state of the whole group consists of the substates of $|H|$ classes of functions and backup server, which is expressed by $\Gamma = (m_1, n_1, o_1, p_1, \cdots, m_{|H|}, n_{|H|}, o_{|H|}, p_{|H|}, q) \in U_j$, where U_j denotes a set of all feasible states for the group containing backup server $j \in S$.

This work shows the analyses for the case of $|H| = 1$ to present the basic idea of estimating the unavailability of a function based on the queueing theory. Then this work shows how to consider the case of $|H| = 2$ and the general case of $|H|$ based on the result of $|H| = 1$.

6.2.1 In case of $|H| = 1$

This work discusses the case of $|H| = 1$, where each element in W has the same average failure rate of λ and the same average repair time of μ^{-1}. A system state of the group containing backup server $j \in S$ is expressed by (m, n, o, p, q).

Number of feasible system states

This work analyzes the number of feasible system states of group containing backup server $j \in S$ in two conditions. One is $|L_j| \geq r_j$, and the other is $|L_j| < r_j$.

When $|L_j| \geq r_j$, we consider two cases; $q = 1$ in case 1 and $q = 0$ in case 2. For case 1, there are two sub cases. In case 1a, $o + p \in [0, r_j - 1]$. Since backup server j has remaining capacity to recover one or more failed functions, $n = 0$ in case 1a. When $o + p = t \in [0, r_j - 1]$, $m = |L_j| - t$ and there are $t + 1$ possibilities for the values of o and p. As a result, in case 1a, there are $\frac{r_j^2 + r_j}{2}$ feasible states. In case 1b, $o + p = r_j$ and $m + n = |L_j| - r_j$. There are $r_j + 1$ and $|L_j| - r_j + 1$ possibilities for the values of o and p and for the values of m and n, respectively. As a result, in case 1b, there are $(r_j + 1)(|L_j| - r_j + 1)$ feasible states. Hence, there are $\frac{r_j^2 + r_j}{2} + (r_j + 1)(|L_j| - r_j + 1)$ feasible states in case 1. For case 2, $o + p = 0$ and $m + n = |L_j|$, which leads to $|L_j| + 1$ feasible states. Therefore, in total $\frac{r_j^2 + r_j}{2} + (r_j + 1)(|L_j| - r_j + 1) + |L_j| + 1$ feasible states exist when $|L_j| \geq r_j$.

When $|L_j| < r_j$, the backup server can recover all functions in L_j at the same time if it is not in the failed state. For the case of $q = 1$, $m + o + p = |L_j|$ and $n = 0$, which indicates $\frac{|L_j|^2 + 3|L_j| + 2}{2}$ feasible states. For the case of $q = 0$, $m + n = |L_j|$ and $o + p = 0$, which leads to $|L_j| + 1$ feasible states. Therefore, in total $\frac{|L_j|^2 + 5|L_j| + 4}{2}$ feasible states exist when $|L_j| < r_j$.

System state transition for (m, n, o, p, q)

Figure 6.3 shows the state transition for state $(m, n, o, p, q) \in U_j$, $j \in S$, where nine types of states can be incoming to state (m, n, o, p, q), which can be outgoing to nine types of states. Table 6.1 describes the condition and transfer rate for each type of transition, where $l = \min(n, r_j)$. This work numbers the

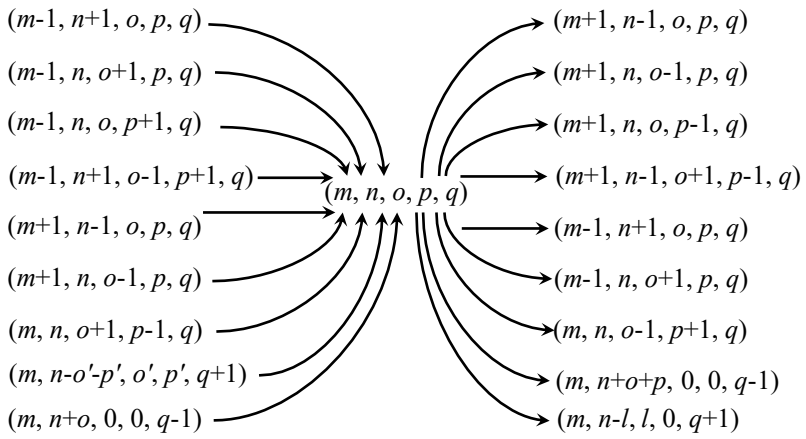


Figure 6.3: System state transition for (m, n, o, p, q).

Table 6.1: System state transition incoming to and outgoing from state (m, n, o, p, q).

Direction	Type, k	State	Transfer rate	Condition
Incoming states	1	$(m-1, n+1, o, p, q)$	$(n+1+o)\mu$	$m \geq 1$ and $o + p = r_j$
				$m \geq 1$ and $q = 0$
	2	$(m-1, n, o+1, p, q)$	$(o+1)\mu$	$m \geq 1$, $o + 1 + p \leq r_j$, and $q = 1$
	3	$(m-1, n, o, p+1, q)$	$(p+1)\mu$	$m \geq 1$, $o + p + 1 \leq r_j$, and $q = 1$
	4	$(m-1, n+1, o-1, p+1, q)$	$(p+1)\mu$	$m \geq 1$, $o \geq 1$, and $o + p = r_j$
	5	$(m+1, n-1, o, p, q)$	$(m+1)\lambda$	$n \geq 1$
	6	$(m+1, n, o-1, p, q)$	$(m+1)\lambda$	$n = 0$ and $o \geq 1$
	7	$(m, n, o+1, p-1, q)$	$(o+1)\delta_j$	$p \geq 1$
	8	$(m, n-o'-p', o', p', q+1)$	λ	$q = 0$
	9	$(m, n+o, 0, 0, q-1)$	μ	$p = 0$ and $q = 1$
Outgoing states	10	$(m+1, n-1, o, p, q)$	$(n+o)\mu$	$n \geq 1$
	11	$(m+1, n, o-1, p, q)$	$o\mu$	$n = 0$ and $o \geq 1$
	12	$(m+1, n, o, p-1, q)$	$p\mu$	$n = 0$ and $p \geq 1$
	13	$(m+1, n-1, o+1, p-1, q)$	$p\mu$	$n \geq 1$ and $p \geq 1$
	14	$(m-1, n+1, o, p, q)$	$m\lambda$	$m \geq 1$ and $o + p = r_j$
				$m \geq 1$ and $q = 0$
	15	$(m-1, n, o+1, p, q)$	$m\lambda$	$m \geq 1$, $o + 1 + p \leq r_j$, and $q = 1$
	16	$(m, n, o-1, p+1, q)$	$o\delta_j$	$o \geq 1$
	17	$(m, n+o+p, 0, 0, q-1)$	λ	$q = 1$
	18	$(m, n-l, l, 0, q+1)$	μ	$q = 0$

types of states from 1 to 18. For type 8 of $(m, n-o'-p', o', p', q+1)$, there are $l+1$ states incoming to state (m, n, o, p, q) with considering the conditions of $o'+p'=l$. Except for type 8, each type corresponds to one state incoming to or outgoing from state (m, n, o, p, q).

This work explains the system state transition incoming to state (m, n, o, p, q), as shown in Fig. 6.3 and Table 6.1, in details:

1. For types 1-4 and 9, the repair procedure is completed for a failed element, function or backup server, that goes to the active state.

    - For type 1, there are two situations. The first situation is that a waiting function goes to the active state with the transfer rate of $(n+1)\mu$; the second situation is that a function in the recovery procedure goes to the active state with the transfer rate of $o\mu$, and a waiting function starts the recovery procedure. Therefore, the total transfer rate for type 1 is $(n+1+o)\mu$. Since there exists at least one waiting function before the transition, or $n+1>0$, we have a condition of $o+p=r_j$ or $q=0$ indicating that the backup server cannot recover any more or any failed function. Note that the first situation can occur in both conditions of $o+p=r_j$ and $q=0$, while the second situation can only occur in the condition of $o+p=r_j$. Since there is at least one function that is in the active state after the transition, we have a condition of $m \geq 1$.
    - For type 2, a function in the recovery procedure goes to the active state with the transfer rate of $(o+1)\mu$, where no waiting function exists. Since the total number of functions that are either recovered or being recovered cannot exceed r_j before the transition, we have a condition of $o+1+p \leq r_j$. The backup server needs to be in the active state to have a function in the recovery procedure, which leads to a conditon of $q=1$. These two conditions include the condition of $n=0$. Similar to type 1, there is a a condition of $m \geq 1$.
    - For type 3, a function that is recovered goes to the active state with the transfer rate of $(p+1)\mu$, where no waiting function exists.

Similar to type 2, we have the conditions of $o + p + 1 \leq r_j$, $q = 1$, and $m \geq 1$.

- For type 4, a function that is recovered goes to the active state with the transfer rate of $(p + 1)\mu$, and a waiting function starts the recovery procedure. Since there exists at least one waiting function before the transition, or $n + 1 > 0$, we have a condition of $o + p = r_j$. Since there are at least one function that is in the active state and at least one function that is in the recovery procedure after the transition, we have the conditions of $m \geq 1$ and $o \geq 1$.
- For type 9, the backup server goes to the active state with the transfer rate of μ, and o waiting functions start the recovery procedure. Since there is no failed function that is recovered before and after the transition, we have a condition of $p = 0$. Clearly, the backup server is in the active state after the transition, which indicates a condition of $q = 1$.

2. For types 5, 6, and 8, a failure occurs at an active element, function or backup server, that goes to the failed state.

   - For type 5, a function goes to the failed state with the transfer rate of $(m + 1)\lambda$, and it starts the waiting procedure. Since there is at lease one waiting function after the transition, we have a condition of $n \geq 1$, which includes the fact that the backup server cannot recover any more or any failed function.
   - For type 6, a function goes to the failed state with the transfer rate of $(m + 1)\lambda$, and it starts the recovery procedure. This function does not experience the waiting procedure, which indicates that no waiting function that fails before the considered function exists, i.e., $n = 0$. Since there is at lease one function in the recovery procedure after the transition, we have a condition of $o \geq 1$.
   - For type 8, the backup server goes to the failed state with the transfer rate of λ, and all failed functions that are either recovered or being recovered go back to the waiting procedure. Clearly, we have a condition of $q = 0$.

3. For type 7, a recovery procedure is completed for a failed function with the transfer rate of $(o+1)\delta_j$. Since there exists at least one function that is recovered after the transition, we have a condition of $p \geq 1$.

The explanations for the state transition outgoing from state (m, n, o, p, q) are similar to the above explanations for the incoming transitions; more specifically, types 1-9 correspond to types 10-18, respectively.

Equilibrium States

Let $P(m, n, o, p, q)$ be the probability that the system is in state $(m, n, o, p, q) \in U_j$, $j \in S$. In the equilibrium state, the total incoming flows to state (m, n, o, p, q) are equivalent to the total outgoing flows from state (m, n, o, p, q). The equilibrium equation for (m, n, o, p, q) is expressed by,

$$\begin{aligned}
&h_1 P(m-1, n+1, o, p, q) + h_2 P(m-1, n, o+1, p, q) + \\
&h_3 P(m-1, n, o, p+1, q) + \\
&h_4 P(m-1, n+1, o-1, p+1, q) + \\
&h_5 P(m+1, n-1, o, p, q) + h_6 P(m+1, n, o-1, p, q) + \\
&h_7 P(m, n, o+1, p-1, q) + h_9 P(m, n+o, 0, 0, q-1) + \\
&h_8 \sum_{o'=0}^{l} P(m, n-l, o', l-o', q+1) \\
&= \sum_{k=10}^{18} h_k P(m, n, o, p, q),
\end{aligned} \tag{6.2}$$

where h_k, $k \in [1, 18]$, equals the transfer rate of type k if the conditions of type k are satisfied and 0 otherwise. The total probability of all states equals one, which indicates an equation as,

$$\sum_{(m,n,o,p,q) \in U_j} P(m, n, o, p, q) = 1. \tag{6.3}$$

By solving the multiple equations, which contain the equilibrium equations for all states and the equation that the sum of all state probabilities equals one, we obtain the equilibrium-state probability of each system state.

Table 6.2: System state transition incoming to and outgoing from state $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$.

Direction	Type	State	Transfer rate	Further condition
Incoming states	1-1	$(m_1-1, n_1+1, o_1, p_1, m_2, n_2, o_2, p_2, q)$	$(n_1+1)\mu_1 + o_1\mu_1 \frac{n_1+1}{n_1+1+n_2}$	$m_1 \geq 1$
	1-2	$(m_1, n_1, o_1, p_1, m_2-1, n_2+1, o_2, p_2, q)$	$(n_2+1)\mu_2 + o_2\mu_2 \frac{n_2+1}{n_1+n_2+1}$	$m_2 \geq 1$
	1-3	$(m_1-1, n_1, o_1+1, p_1, m_2, n_2+1, o_2-1, p_2, q)$	$(o_1+1)\mu_1 \frac{n_2+1}{n_1+n_2+1}$	$m_1 \geq 1$ and $o_2 \geq 1$
	1-4	$(m_1, n_1+1, o_1-1, p_1, m_2-1, n_2, o_2+1, p_2, q)$	$(o_2+1)\mu_2 \frac{n_1+1}{n_1+1+n_2}$	$m_2 \geq 1$ and $o_1 \geq 1$
	2-1	$(m_1-1, n_1, o_1+1, p_1, m_2, n_2, o_2, p_2, q)$	$(o_1+1)\mu_1$	$m_1 \geq 1$
	2-2	$(m_1, n_1, o_1, p_1, m_2-1, n_2, o_2+1, p_2, q)$	$(o_2+1)\mu_2$	$m_2 \geq 1$
	3-1	$(m_1-1, n_1, o_1, p_1+1, m_2, n_2, o_2, p_2, q)$	$(p_1+1)\mu_1$	$m_1 \geq 1$
	3-2	$(m_1, n_1, o_1, p_1, m_2-1, n_2, o_2, p_2+1, q)$	$(p_2+1)\mu_2$	$m_2 \geq 1$
	4-1	$(m_1-1, n_1+1, o_1-1, p_1+1, m_2, n_2, o_2, p_2, q)$	$(p_1+1)\mu_1 \frac{n_1+1}{n_1+1+n_2}$	$m_1 \geq 1$ and $o_1 \geq 1$
	4-2	$(m_1, n_1, o_1, p_1, m_2-1, n_2+1, o_2-1, p_2+1, q)$	$(p_2+1)\mu_2 \frac{n_2+1}{n_1+n_2+1}$	$m_2 \geq 1$ and $o_2 \geq 1$
	4-3	$(m_1-1, n_1, o_1, p_1+1, m_2, n_2+1, o_2-1, p_2, q)$	$(p_1+1)\mu_1 \frac{n_2+1}{n_1+n_2+1}$	$m_1 \geq 1$ and $o_2 \geq 1$
	4-4	$(m_1, n_1+1, o_1-1, p_1, m_2-1, n_2, o_2, p_2+1, q)$	$(p_2+1)\mu_2 \frac{n_1+1}{n_1+1+n_2}$	$m_2 \geq 1$ and $o_1 \geq 1$
	5-1	$(m_1+1, n_1-1, o_1, p_1, m_2, n_2, o_2, p_2, q)$	$(m_1+1)\lambda_1$	$n_1 \geq 1$
	5-2	$(m_1, n_1, o_1, p_1, m_2+1, n_2-1, o_2, p_2, q)$	$(m_2+1)\lambda_2$	$n_2 \geq 1$
	6-1	$(m_1+1, n_1, o_1-1, p_1, m_2, n_2, o_2, p_2, q)$	$(m_1+1)\lambda_1$	$o_1 \geq 1$
	6-2	$(m_1, n_1, o_1, p_1, m_2+1, n_2, o_2-1, p_2, q)$	$(m_2+1)\lambda_2$	$o_2 \geq 1$
	7-1	$(m_1, n_1, o_1+1, p_1-1, m_2, n_2, o_2, p_2, q)$	$(o_1+1)\delta_j$	$p_1 \geq 1$
	7-2	$(m_1, n_1, o_1, p_1, m_2, n_2, o_2+1, p_2-1, q)$	$(o_2+1)\delta_j$	$p_2 \geq 1$
	8	$(m_1, n_1-o_1'-p_1', o_1', p_1', m_2, n_2-o_2'-p_2', o_2', p_2', q+1)$	λ_j^{B}	
	9	$(m_1, n_1+o_1, 0, 0, m_2, n_2+o_2, 0, 0, q-1)$	$\mu_j^{\mathrm{B}} \frac{1}{l_9}$	
Outgoing states	10-1	$(m_1+1, n_1-1, o_1, p_1, m_2, n_2, o_2, p_2, q)$	$n_1\mu_1 + o_1\mu_1 \frac{n_1}{n_1+n_2}$	$n_1 \geq 1$
	10-2	$(m_1, n_1, o_1, p_1, m_2+1, n_2-1, o_2, p_2, q)$	$n_2\mu_2 + o_2\mu_2 \frac{n_2}{n_1+n_2}$	$n_2 \geq 1$
	10-3	$(m_1+1, n_1, o_1-1, p_1, m_2, n_2-1, o_2+1, p_2, q)$	$o_1\mu_1 \frac{n_2}{n_1+n_2}$	$o_1 \geq 1$ and $n_2 \geq 1$
	10-4	$(m_1, n_1-1, o_1+1, p_1, m_2+1, n_2, o_2-1, p_2, q)$	$o_2\mu_2 \frac{n_1}{n_1+n_2}$	$o_2 \geq 1$ and $n_1 \geq 1$
	11-1	$(m_1+1, n_1, o_1-1, p_1, m_2, n_2, o_2, p_2, q)$	$o_1\mu_1$	$o_1 \geq 1$
	11-2	$(m_1, n_1, o_1, p_1, m_2+1, n_2, o_2-1, p_2, q)$	$o_2\mu_2$	$o_2 \geq 1$
	12-1	$(m_1+1, n_1, o_1, p_1-1, m_2, n_2, o_2, p_2, q)$	$p_1\mu_1$	$p_1 \geq 1$
	12-2	$(m_1, n_1, o_1, p_1, m_2+1, n_2, o_2, p_2-1, q)$	$p_2\mu_2$	$p_2 \geq 1$
	13-1	$(m_1+1, n_1-1, o_1+1, p_1-1, m_2, n_2, o_2, p_2, q)$	$p_1\mu_1 \frac{n_1}{n_1+n_2}$	$n_1 \geq 1$ and $p_1 \geq 1$
	13-2	$(m_1, n_1, o_1, p_1, m_2+1, n_2-1, o_2+1, p_2-1, q)$	$p_2\mu_2 \frac{n_2}{n_1+n_2}$	$n_2 \geq 1$ and $p_2 \geq 1$
	13-3	$(m_1+1, n_1, o_1, p_1-1, m_2, n_2-1, o_2+1, p_2, q)$	$p_1\mu_1 \frac{n_2}{n_1+n_2}$	$p_1 \geq 1$ and $n_2 \geq 1$
	13-4	$(m_1, n_1-1, o_1+1, p_1, m_2+1, n_2, o_2, p_2-1, q)$	$p_2\mu_2 \frac{n_1}{n_1+n_2}$	$p_2 \geq 1$ and $n_1 \geq 1$
	14-1	$(m_1-1, n_1+1, o_1, p_1, m_2, n_2, o_2, p_2, q)$	$m_1\lambda_1$	$m_1 \geq 1$
	14-2	$(m_1, n_1, o_1, p_1, m_2-1, n_2+1, o_2, p_2, q)$	$m_2\lambda_2$	$m_2 \geq 1$
	15-1	$(m_1-1, n_1, o_1+1, p_1, m_2, n_2, o_2, p_2, q)$	$m_1\lambda_1$	$m_1 \geq 1$
	15-2	$(m_1, n_1, o_1, p_1, m_2-1, n_2, o_2+1, p_2, q)$	$m_2\lambda_2$	$m_2 \geq 1$
	16-1	$(m_1, n_1, o_1-1, p_1+1, m_2, n_2, o_2, p_2, q)$	$o_1\delta_j$	$o_1 \geq 1$
	16-2	$(m_1, n_1, o_1, p_1, m_2, n_2, o_2-1, p_2+1, q)$	$o_2\delta_j$	$o_2 \geq 1$
	17	$(m_1, n_1+o_1+p_1, 0, 0, m_2, n_2+o_2+p_2, 0, 0, q-1)$	λ_j^{B}	
	18	$(m_1, n_1-o_1', o_1', 0, m_2, n_2-o_2', o_2', 0, q+1)$	$\mu_j^{\mathrm{B}} \frac{1}{l_{18}}$	

Estimate unavailability of function

For state $(m, n, o, p, q) \in U_j$, $j \in S$, there are $n+o$ functions that are unavailable. The average number of unavailable functions in the group is computed by $\sum_{(m,n,o,p,q)\in U_j}(n+o)P(m,n,o,p,q)$. For types 14 and 15, the transition outgoing from state (m, n, o, p, q) to its corresponding state indicates that a function becomes unavailable with the rate of $m\lambda$. For type 17, the transition indicates that a backup server becomes to fail and p functions become unavailable with the rate of λ. Hence, per unite time, $\sum_{(m,n,o,p,q)\in U_j}(m\lambda + pq\mu)P(m,n,o,p,q)$ functions become unavailable. Therefore, by using Little's formula [116], the average unavailable time of function $i \in L_j$ is expressed by,

$$T_i = \frac{\sum_{(m,n,o,p,q)\in U_j}(n+o)P(m,n,o,p,q)}{\sum_{(m,n,o,p,q)\in U_j}(m\lambda + pq\lambda)P(m,n,o,p,q)}. \tag{6.4}$$

Similarly, the average available time of function i in the group is expressed by,

$$T_i' = \frac{\sum_{(m,n,o,p,q)\in U_j}(m+p)P(m,n,o,p,q)}{\sum_{(m,n,o,p,q)\in U_j}[(n+o+p)\mu + o\delta_j)]P(m,n,o,p,q)}. \tag{6.5}$$

Based on (6.4) and (6.5), we obtain the unavailability of function $i \in L_j, j \in S$, by using (6.1).

6.2.2 In case of $|H| = 2$

This work discusses the case that there are two classes of average failure rate; a system state of a group is expressed by $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$. Section 6.2.1 indicates that one key point of the queueing analysis is to specify the condition and transfer rate for each type of transition of a state, which is presented as Table 6.1 for the case of $|H| = 1$. This work shows how to achieve this point in the case of $|H| = 2$ by using the result of $|H| = 1$.

State $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$ can be viewed as state (m, n, o, p, q), where $m = m_1 + m_2, n = n_1 + n_2, o = o_1 + o_2$, and $p = p_1 + p_2$. The types of transition in the case of $|H| = 2$ are the same with those in Table 6.1 for the case of $|H| = 1$. The conditions of a type of transition for state (m, n, o, p, q) in Table 6.1 are included in the conditions of the same transition type for state $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$. Now, for each transition type, a change of function state in each of its corresponding situations can occur at different classes

of functions, which leads to several possible states. Based on the state column in Table 6.1, this work specifies each possible state, the corresponding transfer rate, and the further conditions in addition to those in Table 6.1 for each type of transition incoming to or outgoing from state $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$.

Table 6.2 shows the possible states, transfer rate, and further conditions for all types of transition for state $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$, where λ_j^{B} and μ_j^{B} denote the average failure rate and average repair rate of backup server j, respectively. The transition of type 8 corresponds to l_8 states, where l_8 is the number of combinations of o_1', p_1', o_2', and p_2' with satisfying the conditions of $o_1' + p_1' \leq n_1$, $o_2' + p_2' \leq n_2$, and $o_1' + o_2' + p_1' + p_2' = l$. For the transition of type 9 with state $(m_1, n_1 + o_1, 0, 0, m_2, n_2 + o_2, 0, 0, q - 1)$, when the backup server becomes active, the system state is outgoing to one of the l_9 states with the form of $(m_1, n_1 + o_1 - o_1', o_1', 0, m_2, n_2 + o_2 - o_2', o_2', 0, q)$, where l_9 is the number of combinations of o_1' and o_2' with satisfying the conditions of $o_1' + o_2' \leq r_j$, $o_1' \leq n_1 + o_1$, $o_2' \leq n_2 + o_2$, and, when $o_1' + o_2' < r_j$, $n_1 + o_1 - o_1' + n_2 + o_2 - o_2' = 0$. Similarly, for the transition of type 18, state $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$ is outgoing to one of the l_{18} states with the form of $(m_1, n_1 - o_1', o_1', 0, m_2, n_2 - o_2', o_2', 0, q + 1)$, where l_{18} is the number of combinations of o_1' and o_2' with satisfying the conditions of $o_1' + o_2' = l$, $o_1' \leq n_1$, and $o_2' \leq n_2$.

This work takes the transition of type 1 for state $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$ in Table 6.2 as an example to explain in details. There are two situations for the transition of type 1-1 with state $(m_1 - 1, n_1 + 1, o_1, p_1, m_2, n_2, o_2, p_2, q)$. In the first situation, one of the $n_1 + 1$ functions becomes active, which indicates that the state is outgoing to state $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$ with the rate of $(n_1 + 1)\mu_1$. In the second situation, where one of the o_1 functions becomes active, one of waiting functions is randomly chosen to start the recovery procedure, which leads to two cases of states which state $(m_1 - 1, n_1 + 1, o_1, p_1, m_2, n_2, o_2, p_2, q)$ is outgoing to. In one case, a function in class 1 starts the recovery procedure and the state is outgoing to state $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$. In the other case, a function in class 2 starts the recovery procedure and the state is outgoing to state $(m_1, n_1 + 1, o_1 - 1, p_1, m_2, n_2 - 1, o_2 + 1, p_2, q)$. Therefore, the transfer rate that the state is outgoing to $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$ is $(n_1 + 1)\mu_1 + o_1\mu_1 \frac{n_1+1}{n_1+1+n_2}$, where $\frac{n_1+1}{n_1+1+n_2}$ is the probability that the function starting the recovery procedure in the second situation is in class 1. For the transition of type 1-3 with

state $(m_1 - 1, n_1, o_1 + 1, p_1, m_2, n_2 + 1, o_2 - 1, p_2, q)$, when one of the $o_1 + 1$ functions becomes active, if one of the n_1 functions starts the recovery procedure, the state is outgoing to state $(m_1, n_1 - 1, o_1 + 1, p_1, m_2, n_2 + 1, o_2 - 1, p_2, q)$; otherwise, the state is outgoing to state $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$. Therefore, the transfer rate that the state is outgoing to $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q)$ is $(o_1 + 1)\mu_1 \frac{n_2+1}{n_1+n_2+1}$, where $\frac{n_2+1}{n_1+n_2+1}$ is the probability that the function starting the recovery procedure is in class 2. The transitions of types 1-2 and 1-4 are similar to the transitions of types 1-1 and 1-3, respectively.

Based on the condition and transfer rate for each type of transition, we obtain the state probability for $(m_1, n_1, o_1, p_1, m_2, n_2, o_2, p_2, q) \in U_j, j \in S$, through the same technique introduced in Section 6.2.1. By applying (6.4) and (6.5) for each class of functions, we obtain the unavailability of each function in a group.

Similarly, we can deal with the general case of $|H|$ by adopting the above procedure. In the general case, the average unavailable and available times of function $i \in L_j, j \in S$, with class $h \in H$ are computed by,

$$T_i = \frac{\sum_{\Gamma \in U_j} (n_h + o_h) P(\Gamma)}{\sum_{\Gamma \in U_j} (m_h \lambda_h + p_h q \lambda_j^{\mathrm{B}}) P(\Gamma)} \tag{6.6}$$

and

$$T_i' = \frac{\sum_{\Gamma \in U_j} (m_h + p_h) P(\Gamma)}{\sum_{\Gamma \in U_j} [(n_h + o_h + p_h)\mu_h + o_h \delta_j)] P(\Gamma)}, \tag{6.7}$$

respectively.

6.3 Heuristic algorithm

This work introduces a heuristic algorithm based on SA [91] (see Algorithm 11) to solve the UASBA problem. Let D^{init}, D^{term}, and ρ denote three given parameters in the SA algorithm, where $D^{\mathrm{init}} > D^{\mathrm{term}} > 0$ and $0 < \rho < 1$; D represents a running variable to indicate the "temperature" in SA. At the beginning of SA algorithm, $D = D^{\mathrm{init}}$ and a feasible solution consisting of $x_{ij}, i \in F, j \in S$ is randomly set; the maximum unavailability among functions in each feasible solution is computed by the approach provided in Section 6.2.

Then, in each iteration, $D = \rho D$ and a new feasible solution is generated by applying a random change to the existing solution. If the objective value of new solution is less than that of the existing one, the new solution is accepted and replaces the existing one; otherwise, the new solution is accepted with a probability, which depends on the difference between the objective values of new and existing solutions and the value of D. As a result, the SA algorithm can avoid a local minimum solution. The algorithm is terminated when $D \leq D^{\text{term}}$.

Algorithm 11: Simulated annealing (SA)

Input: λ_h and μ_h, $h \in H$, for function $i \in F$ and backup server $j \in S$, δ_j, D^{init}, D^{term}, and ρ

Ouput: x_{ij}, Q_i, and J

Set $D = D^{\text{init}}$

Randomly generate feasible solution of x_{ij}

Compute Q_i and J for x_{ij}

while $D > D^{term}$ **do**

  Set $D = \rho D$

  Generate new feasible solution of x'_{ij}

  Compute Q'_i and J' for x'_{ij}

  Set $x_{ij} = x'_{ij}$, $Q_i = Q'_i$, and $J = J'$ with a probability of $\min(1, d)$, where $d = e^{(\frac{J-J'}{D})}$

end

This work analyzes the computational time complexity of SA algorithm. It takes $O(|F||S|)$ to generate a feasible solution. Given a feasible solution, as introduced in Section 6.2, this work creates and solves the multiple linear equations for each group to obtain the maximum unavailability. Consider the maximum number of feasible states, or the maximum number of variables in multiple equations, among all groups as $O(A)$; $A = r^2$ in the case of $|H| = 1$, where $r = \max_{j \in S} r_j$. The computational time complexity of creating the multiple equations for a group is $O(A)$. Solving the multiple equations for a group takes $O(A^2)$ when the Gauss-Seidel algorithm [117] is adopted. Therefore, computing the maximum unavailability for a given solution takes $O(A^2|S|)$, which is the computational time complexity for each iteration of the SA algorithm.

For the approximation performance of SA algorithm, there is a tradeoff between the number of iterations and the accuracy of solution. A more accurate solution can be obtained by increasing the number of iterations, or increasing the value of ρ.

6.4 Numerical results

This work first introduces a conventional model as a baseline model. Then this work compares the proposed model with the baseline model in terms of the maximum unavailability among functions. The performance dependencies on the backup server capacity, failure rate, average repair time, and average recovery time are evaluated. This work uses Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory through the evaluations.

6.4.1 Baseline model

Without comprehensively considering the failure, repair, recovery, and waiting behaviors of functions and backup servers, the conventional backup allocation models with shared protection are not explicitly aware of unavailability. Instead of considering the function unavailability directly, the work in [19] presented a survivability-aware backup allocation model, which is considered as the baseline model.

In the baseline model, element $k \in W$ with class $h \in H$ is associated with a failure probability, which is considered as $\alpha_k = \frac{\mu_h^{-1}}{\mu_h^{-1}+\lambda_h^{-1}}$. Let $\Lambda \subseteq F$ denote a set of failed functions. The survivability for a backup allocation, which is denoted by K, is defined as the probability that the protection for all functions in Λ succeed. The baseline model aims to find the backup allocation that maximizes the survivability.

As studied in [19], the survivability is separately considered for each connected component, which is either a group containing functions and the preassigned backup server or a single function. Let C denote the set of all connected components; K_u represents the survivability of connected component $u \in C$. For the connected component of group containing backup server $j \in S$, $K_u = \alpha_j \Pi_{i \in L_j}(1-\alpha_i) + (1-\alpha_j) \sum_{\Lambda \in L_j : |\Lambda| \leq r_j} \Pi_{i' \in \Lambda} \alpha_{i'} \Pi_{i \in L_j \setminus \Lambda}(1-\alpha_i)$. For the con-

nected component of single function $i \in F$, $K_u = 1-\alpha_i$. The overall survivability is computed by, $K = \Pi_{u \in C} K_u$.

This work solves the baseline model by using the SA algorithm introduced in Section 6.3, where the objective is changed to maximize the overall survivability and d is modified to $e^{(\frac{K'-K}{D})}$. After obtaining the result of backup allocation, the exact maximum unavailability among functions is computed by the approach provided in Section 6.2.

6.4.2 Experiment Setup

A network with 100 middleboxes, or $|F| = 100$, is considered in the experiments. The number of backup servers is considered as $|S| = 20$. Each result is an average over 500 trials, in each of which this work randomly selects two integers from the range of $[1, M]$ as the values of c_j and r_j, $j \in S$, respectively, with considering the condition of $c_j \geq r_j$, where M denotes the upper bound of range. Similarly, the average recovery time of δ_j^{-1} on backup server $j \in S$ is randomly set within a range of $[L, N]$ in each trial, where L and N denote the lower and upper bounds of range, respectively. This work sets $D^{\text{init}} = 10^7$, $D^{\text{term}} = 10^{-5}$, and $\rho = 0.99$ for the SA algorithm.

According to the studies in [22], the median and 95th percentile of time between two adjacent failures for devices in a network are 10^4 and 10^6 [s], respectively; the median and 95th percentile of repair time are 10^3 and 10^5 [s], respectively. In the experiments in [20], the recovery time on a backup server varies from 30 to 10^2 [s]. In the evaluations, the parameters are set with refereeing the above investigations.

6.4.3 Evaluation for $|H| = 1$

Figure 6.4 shows the maximum unavailabilities obtained by the proposed and baseline models for different values of upper bound of capacity range of backup servers, where $\lambda = 10^{-4}$, $\mu^{-1} = 10^3$, and the range of δ_j^{-1}, $j \in S$, is considered as $[30, 10^2]$. This work observes that, as the value of M increases, the maximum unavailability obtained by the proposed model decreases. This is because, with a larger value of r_j, $j \in S$, a protected function can have less probability to be in the waiting procedure. Compared to the baseline model, the proposed

unavailability-aware model reduces the maximum unavailability 9% in average.

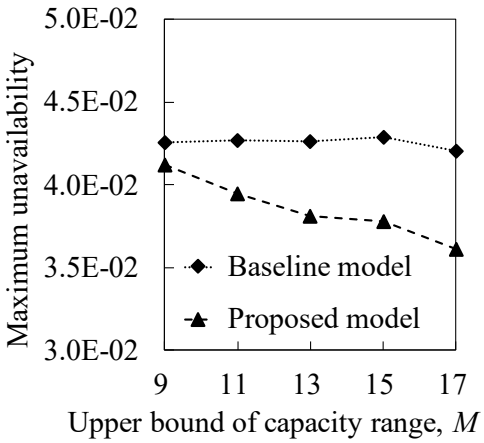


Figure 6.4: Comparison among maximum unavailabilities obtained by different models for different values of M.

Table 6.3 shows the computation times of proposed and baseline models to obtain the results shown in Fig. 6.4. The computation time of proposed model increases as the value of M increases, which fits the theoretical analyses in Section 6.3. The results observe that the computation time of proposed model is 13 times longer in average than that of baseline model. However, since it is much smaller than the investigated time between two adjacent failures, repair time, and recovery time, the proposed model can be acceptable in practical applications in terms of the computation time.

Table 6.3: Computation time [s] to obtain results shown in Fig. 6.4.

M	Proposed model	Baseline model
9	1.288	0.133
11	1.531	0.129
13	1.834	0.135
15	2.098	0.139
17	2.390	0.143

Figure 6.5 shows the dependency of maximum unavailabilities obtained by

the proposed and baseline models on the average failure rate, where $M = 15$, $\mu^{-1} = 10^3$, and the range of $\delta_j^{-1}, j \in S$, is considered as $[30, 10^2]$. As the average failure rate increases, the maximum unavailabilities of both models increase. The proposed model reduces the maximum unavailability 11% in average compared to the baseline model.

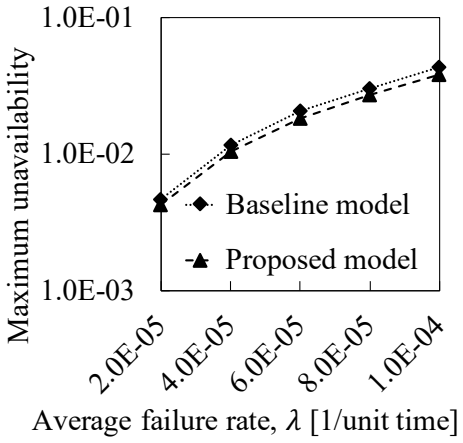


Figure 6.5: Dependency of maximum unavailabilities obtained by different models on average failure rate.

Figure 6.6 compares the maximum unavailabilities obtained by the proposed and baseline models for different values of average repair time, where $M = 15$, $\lambda = 10^{-4}$, and the range of $\delta_j^{-1}, j \in S$, is considered as $[30, 10^2]$. This work observes that the maximum unavailabilities of both models increase as the average repair time increases. On one hand, a larger average repair time for a backup server leads to a higher probability that a failed function is in the waiting procedure. On the other hand, the time that a failed function occupies a backup server becomes long as the average repair time of a function increases. As a result, the probability that a failed function is in the waiting procedure becomes high. The proposed model reduces the maximum unavailability 10% in average compared to the baseline model.

Figure 6.7 shows the comparison among maximum unavailabilities obtained by the proposed and baseline models for different values of upper bound of repair time range, where the lower bound is set to $L = 30$, $M = 15$, $\lambda = 10^{-4}$, and $\mu^{-1} = 10^3$. This work observes that, as the value of N increases, it takes

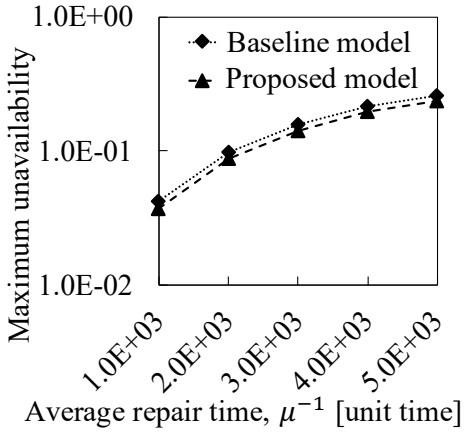


Figure 6.6: Comparison among maximum unavailabilities obtained by different models for different values of average repair times.

longer time to recover a failed function, which increases the maximum unavailabilities for both models. In this evaluation, compared to the baseline model, the proposed model reduces the maximum unavailability 12% in average.

6.4.4 Evaluation for $|H| = 2$

This work considers that there are two typical types of elements in the network. The first type of element has a higher probability to fail and needs a longer time to be repaired than the second type of element, or $\lambda_1 \geq \lambda_2$ and $\mu_1^{-1} \geq \mu_2^{-1}$. For example, an element with the first type may have been used for a longer time than that with the second type. For elements in W, let v denote the ratio of the number of elements with the first type to the total number of elements. This work sets $\lambda_2 = 2 \times 10^{-5}$, $\mu_2^{-1} = 10^3$, and $M = 10$; the range of $\delta_j^{-1}, j \in S$, is considered as $[30, 10^2]$.

Figure 6.8 compares the maximum unavailabilities obtained by the proposed and baseline models for different values of v, where $\lambda_1 = 10^{-4}$ and $\mu_1^{-1} = 5 \times 10^3$. As the value of v increases, the maximum unavailabilities for both models increase. This is because the more the number of functions with the higher failure rate of λ_1 is, the more the average number of failed functions is, which increases the average waiting time for a failed function to start the

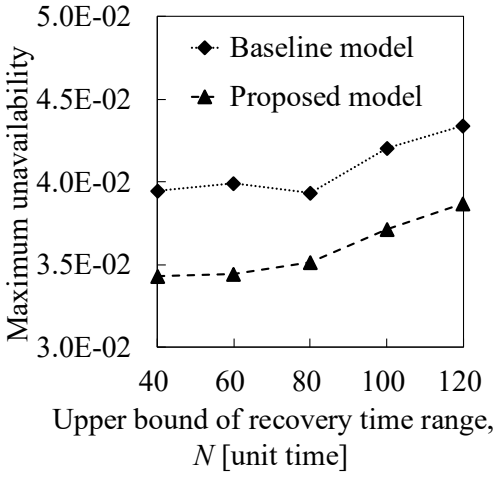


Figure 6.7: Comparison among maximum unavailabilities obtained by different models for different values of N.

recovery procedure. In this evaluation, compared to the baseline model, the proposed model reduces the maximum unavailability 25% in average.

Table 6.4 shows the computation times of proposed and baseline models to obtain the results shown in Fig. 6.8. This work observes that, as v increases from 0.1, the computation time for each model increases. After one point, or $v = 0.5$, as v increases, the computation time for each model decreases. This is because, as discussed in Section 6.3, the computation time of each iteration in SA is proportional to the number of feasible system states, which achieves the maximum value when the difference between the numbers of functions in the two types is minimized.

Table 6.4: Computation time [s] to obtain results shown in Fig. 6.8.

v	Proposed model	Baseline model
0.1	12.853	0.557
0.3	25.943	0.694
0.5	28.770	0.735
0.7	24.670	0.657
0.9	11.935	0.516

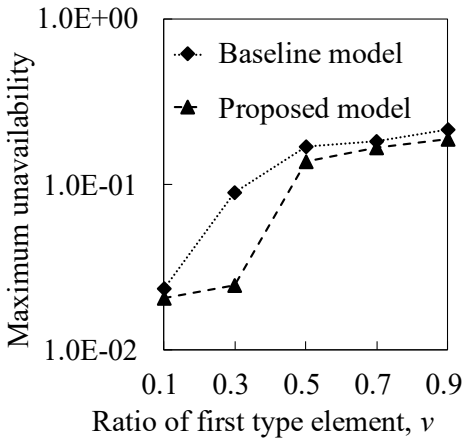


Figure 6.8: Comparison among maximum unavailabilities obtained by different models for different values of v.

Figure 6.9 presents the maximum unavailabilities obtained by the proposed and baseline models for different values of λ_1, where $\mu_1^{-1} = 5 \times 10^3$ and $v = 0.5$. This work observes that, as the value of λ_1 increases, in average more failed functions need to be recovered on the backup server, which increases the average waiting time for a failed function. As a result, the maximum unavailabilities for both models increase. The proposed model reduces the maximum unavailability 22% in average compared to the baseline model.

Figure 6.10 compares the maximum unavailabilities obtained by the proposed and baseline models for different values of μ_1^{-1}, where $\lambda_1 = 10^{-4}$ and $v = 0.5$. This work observes that, as the value of μ_1 increases, with the similar reasons provided for Fig. 6.6, the probability that a failed function is in the waiting procedure becomes high, which increases the maximum unavailability obtained by each of the two models. Compared to the baseline model, the maximum unavailability is reduced 20% in average by adopting the proposed model.

In addition, this work observes that the benefit of proposed model is more significant when $|H| = 2$ compared to the results obtained for $|H| = 1$, where the average reduction is increased from 11% to 22%. This is because, as $|H|$ increases, the number of combinations, or feasible solutions, for backup allocation increases, and the difference between two feasible solutions in terms of the

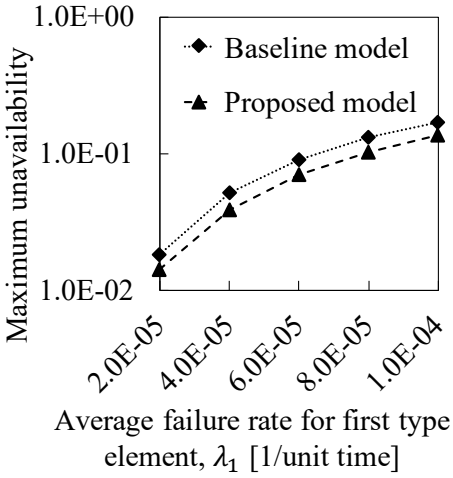


Figure 6.9: Comparison among maximum unavailabilities obtained by different models for different values of λ_1.

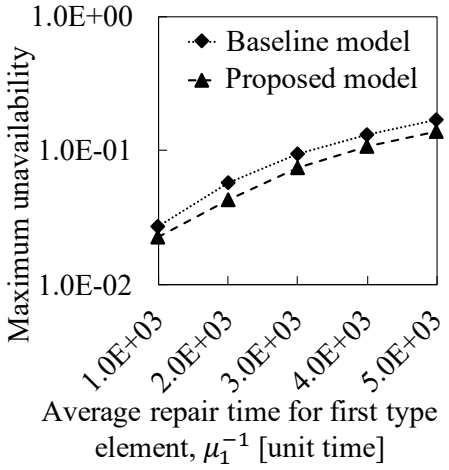


Figure 6.10: Comparison among maximum unavailabilities obtained by different models for different values of μ_1^{-1}.

maximum unavailability can increase; the proposed model which is explicitly aware of the middlebox unavailability can have a greater possibility to find a better solution than the baseline model.

6.5 Chapter summary

This chapter proposed an unavailability-aware backup allocation model with the shared protection for middleboxes with comprehensively considering heterogeneous procedures of functions and backup servers. The backup resources on a backup server can be shared by multiple functions. The proposed model aims to find the backup allocation for all functions to minimize the maximum unavailability among functions. This work developed an analytical approach based on the queueing theory to estimate the unavailability of middleboxes for a given backup allocation. The heterogeneous failure, repair, recovery, and waiting procedures of functions and backup servers, which lead to several different states for each function and for the whole system, are considered in the queueing approach. This work analyzed what all system states are, how they transit from/to each other, and what the equilibrium-state probability of each system state is. Based on the analytical approach, the SA heuristic was introduced to solve the backup allocation problem. The performance dependencies on the backup server capacity, failure rate, average repair time, and average recovery time were evaluated. This work compared the proposed model with the baseline model. The results observed that, compared to the baseline model, the proposed unavailability-aware model reduces the maximum unavailability 16% in average in our examined scenarios.

Chapter 7

Master and slave controller assignment model

This thesis proposes a master and slave controller assignment model against multiple controller failures with considering propagation latency between controllers and switches. A part of the work in this chapter was presented in [71]. In the proposed model, a set of controllers, each of which has a capacity and failure probability, is assigned to each switch to satisfy its survivability guarantee. The propagation latency for a switch depends on the assigned controllers, their failure probabilities, and the selection for the master controller in each failure case. This work defines the average-case expected propagation latency, the worst-case expected propagation latency, and the expected number of switches within a propagation latency bound, as three different objectives to be optimized, which lead to three different problems. Given a set of assigned controllers, the master controller in each failure case is determined by adopting a policy-based approach. This work proves that a low latency first policy (LLFP), in which a controller with lower propagation latency has a higher priority to be used as a master controller, achieves the optimal objectives in the considered problems.

This work formulates the proposed master and slave controller assignment model with different goals as three MILP problems. This work analyzes the considered problems by proving that all the three problems are NP-complete and by showing how they are different from a classical weighted bipartite *b*-

matching (WBM) problem. Based on the analyses, this work introduces a greedy algorithm with polynomial time complexity. This work shows that one of the considered objective functions is a monotone submodular function; the introduced algorithm achieves a 1/2-approximation for the case without the survivability guarantee constraint. This work presents the competitive evaluation in terms of the objective values and the computation times among the results obtained by the proposed model with different approaches and a baseline. The numerical results reveal that the proposed model obtains the optimal objective value with the computation time about 10^2 times shorter than that of the baseline by adopting the policy-based approach with LLFP. Furthermore, when the problem size is small, the introduced heuristic obtains the solution, where the difference between the obtained objective value and the optimal value is less than 14% of the optimal value, with the computation time about 10^3 times shorter than that of solving the MILP problem. As the problem size increases, the heuristic provides a better objective value than the MILP problem even when the admissible computation time to solve the MILP problem is set to 10^3 times longer than that required by the heuristic.

The rest of this chapter is organized as follows. Section 7.1 presents the proposed optimization model for the master and slave controller assignment problems. Section 7.2 analyzes the optimality of LLFP. The proof of NP-completeness is provided in Section 7.3. Section 7.4 introduces the heuristic algorithm. The performance of proposed model is evaluated in Section 7.5. Section 7.6 summaries this chapter.

7.1 Optimization model

Consider a set of switches and a set of controllers, which are denoted by S and C with the sizes of $|S|$ and $|C|$, respectively, in an SDN. Given the placement of switches and controllers in the SDN, the propagation latency between switch $i \in S$ and controller $j \in C$ is represented by l_{ij}. The maximum number of switches hosted by controller j is represented by c_j. The failure probability of controller j is represented by p_j; this work assumes that each controller fails independently. Let q_i denote the acceptable unavailability of switch i. Table 7.1 summarizes the frequently used notations.

Table 7.1: List of frequently used notations in Chapter 7.

Notations	Meaning
l_{ij}	Given parameter indicating propagation latency between switch $i \in S$ and controller $j \in C$
q_i	Given parameter indicating acceptable unavailability of switch i
b_i	Given parameter indicating propagation latency bound of switch i
p_j	Given parameter indicating failure probability of controller j
c_j	Given parameter indicating capacity of controller j
y_{ij}	Given parameter indicating priority of controller j to become master controller of switch i
x_{ij}	Binary variable indicating whether controller j is assigned to switch i

7.1.1 Assign master and slave controllers to switch

Let $x_{ij}, i \in S, j \in C$, denote a binary decision variable; x_{ij} is set to one if controller j is assigned to switch i and zero otherwise. In this model, a set of controllers, which is denoted by $C_i = \{j | j \in C : x_{ij} = 1\} \subseteq C$, is assigned to switch i to satisfy the requirement of acceptable unavailability, before any failure occurs. Let H represent a set of pairs of $(i, j), i \in S, j \in C$, each of which does not allow $x_{ij} = 1$. For example, if the connection between switch i and controller j is not satisfied with the management requirements, such as the admissible delay of connection setup, $x_{ij} = 1$ is not allowed.

At any time, one of controllers in C_i works as the master controller to control switch $i \in S$; others are slave controllers. Once the existing master controller fails, one of the available slave controllers becomes the new master controller to promptly recover the control. Switch i becomes unavailable when all controllers in C_i fail. The survivability guarantee of switch i is expressed by,

$$\prod_{j \in C_i} p_j \leq q_i. \tag{7.1}$$

Figure 7.1 presents an example of master and slave controller assignment with considering the given parameters shown in Fig. 7.1(a). From Fig. 7.1(b), for each controller, the number of assigned switches does not exceed its capacity of c_j; for each switch, a set of controllers is assigned such that the survivability guarantee in (7.1) is satisfied.

l_{ij} [μs], $i \in S, j \in C$

i \ j	1	2	3
1	10	10^4	10^4
2	10^3	10^5	10
3	10^2	10^5	10^4
4	10^4	10^4	10^5
5	10	10^3	10^5
6	10	10	10^5

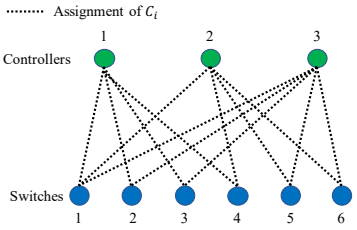
$c_j, p_j, j \in C$

j	1	2	3
c_j	4	6	8
p_j	10^{-3}	10^{-2}	10^{-1}

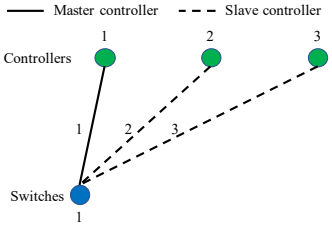
$q_i, i \in S$

i	1	2	3	4
q_i	10^{-4}	10^{-4}	10^{-3}	10

(a) Values of given parameters



(b) Assignment of C_i



(c) Master and slave controllers for switch 1

Figure 7.1: Example of master and slave controller assignment.

7.1.2 Priority policy

Given an assignment of $C_i, i \in S$, the master controller in each failure case is determined by adopting a policy-based approach. Let y_{ij}, $i \in S, j \in C$, denote a given parameter, which indicates that controller j is with the y_{ij}th highest priority over all the controllers in C to become the master controller of switch i, where $1 \leq y_{ij} \leq |C|$. The controllers in C_i are sorted in an increasing order of y_{ij}. At any time, the first, i.e., highest-prioritized, available controller in C_i works as a master controller to hold the workloads on switch i; other available controllers in C_i are slave controllers. When the existing master controller in C_i fails, switch i is transferred to the first, i.e., highest-prioritized, available slave controller in C_i, which becomes the new master controller for switch i. Since the master controller in each failure case is automatically decided based on the policy before any failure occurs, a prompt recovery is provided.

Considering that, at any time, the propagation latency for a switch should be as low as possible to achieve a high network performance in terms of data transmission, this work adopts LLFP as the priority policy, which means that

a controller with lower propagation latency has a higher priority to be used as a master controller, or $y_{ik} < y_{ij}$, if $l_{ik} \leq l_{ij}$, $k, j(k \neq j) \in C_i$. Figure 7.1(c) specifies the master and slave controllers for switch 1 based on LLFP and the assignment shown in Fig. 7.1(b) when all the three controllers are available; the number on each link indicates the priority of each controller. Later, this work will show that LLFP achieves the optimal objectives in the considered problems compared to other policies.

7.1.3 Minimize average-case expected propagation latency

Each switch is only controlled by its master controller. The expected propagation latency for switch, L_i^{E}, is defined by,

$$L_i^{\mathrm{E}} = \sum_{j \in C: x_{ij}=1} \left[l_{ij} \prod_{k \in C: x_{ik}=1, y_{ik}<y_{ij}} p_k(1-p_j) \right], \forall i \in S. \tag{7.2}$$

Equation (7.2) indicates that controller $j \in C_i, i \in S$ becomes the master controller for switch i if all the controllers that are with smaller orders than controller j in C_i fail simultaneously and controller j does not fail. The average-case expected propagation latency, which is denoted by A, is defined by,

$$A = \frac{1}{|S|} \sum_{i \in S} L_i^{\mathrm{E}}. \tag{7.3}$$

The master and slave controller assignment to minimize the average-case expected propagation latency (MinAEL) problem is obtained as the following optimization problem.

$$\begin{aligned}
\min \quad & \frac{1}{|S|} \sum_{i \in S} L_i^{\mathrm{E}} && \text{(7.4a)} \\
\text{s.t.} \quad & (7.2) && \text{(7.4b)} \\
& \sum_{i \in S} x_{ij} \leq c_j, \forall j \in C && \text{(7.4c)} \\
& \sum_{j \in C} x_{ij} \log p_j \leq \log q_i, \forall i \in S && \text{(7.4d)} \\
& x_{ij} = 0, \forall (i,j) \in H && \text{(7.4e)}
\end{aligned}$$

$$x_{ij} \in \{0,1\}, \forall i \in S, j \in C. \tag{7.4f}$$

The objective function in (7.4a) minimizes the average-case expected propagation latency between switches and controllers. Equation (7.4c) indicates that controller $j \in C$ controls at most c_j number of switches. This work takes the logarithmic for both sides of (7.1) to express it as a linear form in (7.4d).

Let $N_i = \{n_{ij} | n_{ij}, j \in C\}$, $i \in S$, where $n_{ij} \in [0,1]$ is the number of assignments between controller j and switch i; $\mathbb{N}_i$ is a set of N_i. If N_i is given, we obtain the expected propagation latency for switch $i \in S$, L_i^{E}, by using (7.2) that is expressed by $L_i^{\mathrm{E}} = \Omega(N_i)$. Let $t_{ij}^{n_{ij}}$, $i \in S$, $j \in C$, $n_{ij} \in [0,1]$, denote a binary variable that is set to one if there is n_{ij} assignment between switch i and controller j, and zero otherwise. In other words, $t_{ij}^0 = 0$ and $t_{ij}^1 = 1$ when controller j is assigned to switch i; $t_{ij}^0 = 1$ and $t_{ij}^1 = 0$ when controller j is not assigned to switch i. Equations (7.4a)-(7.4f) are transformed to the following optimization problem.

$$\min \quad \frac{1}{|S|} \sum_{i \in S} L_i^{\mathrm{E}} \tag{7.5a}$$

$$\text{s.t.} \quad (7.4c) - (7.4f) \tag{7.5b}$$

$$\sum_{n_{ij}=0}^{1} t_{ij}^{n_{ij}} = 1, \forall i \in S, j \in C \tag{7.5c}$$

$$x_{ij} = \sum_{n_{ij}=0}^{1} n_{ij} t_{ij}^{n_{ij}}, \forall i \in S, j \in C \tag{7.5d}$$

$$L_i^{\mathrm{E}} \geq \sum_{N_i \in \mathbb{N}_i} \left[\Omega(N_i) \prod_{j \in C} t_{ij}^{n_{ij}} \right], \forall i \in S \tag{7.5e}$$

$$t_{ij}^{n_{ij}} \in \{0,1\}, \forall i \in S, j \in C, n_{ij} \in [0,1]. \tag{7.5f}$$

Equation (7.5c) indicates that controller $j \in C$ is either assigned to switch $i \in S$ or not assigned to i. Equation (7.5d) shows the relationship between x_{ij} and $t_{ij}^{n_{ij}}$, $n_{ij} \in [0,1]$. Equation (7.5e) indicates that assignment $N_i \in \mathbb{N}_i, i \in S$, is selected to obtain L_i^{E} if all corresponding $t_{ij}^{n_{ij}}$, $j \in C$, of N_i are determined to be 1.

This work expresses (7.5e) in a linear form by introducing some binary variables, $z_{ik}^{N_i}$, $N_i \in \mathbb{N}_i$, $i \in S$, $k \in [1, |C|]$, where $z_{i1}^{N_i} = t_{i1}^{n_{i1}}$ and $z_{ik}^{N_i} = z_{i,k-1}^{N_i} t_{ik}^{n_{ik}}$ if

$k > 1$. Thus, $\prod_{j \in C} t_{ij}^{n_{ij}} = z_{i|C|}^{N_i}$. The equation of $z_{ik}^{N_i} = z_{i,k-1}^{N_i} t_{ik}^{n_{ik}}$ is expressed as a linear form in the following.

$$z_{ik}^{N_i} \leq z_{i,k-1}^{N_i}, \forall N_i \in \mathbb{N}_i, i \in S, k \in [2, |C|] \tag{7.6a}$$

$$z_{ik}^{N_i} \leq t_{ik}^{n_{ik}}, \forall N_i \in \mathbb{N}_i, i \in S, k \in [2, |C|] \tag{7.6b}$$

$$z_{ik}^{N_i} \geq z_{i,k-1}^{N_i} + t_{ik}^{n_{ik}} - 1, \forall N_i \in \mathbb{N}_i, i \in S, k \in [2, |C|] \tag{7.6c}$$

$$z_{ik}^{N_i} \in \{0, 1\}, \forall N_i \in \mathbb{N}_i, i \in S, k \in [1, |C|]. \tag{7.6d}$$

Equations (7.5a)-(7.5f) are transformed to the following MILP problem.

$$\min \quad \frac{1}{|S|} \sum_{i \in S} L_i^{\mathrm{E}} \tag{7.7a}$$

$$\text{s.t.} \quad (7.4c) - (7.4f), (7.5c), (7.5d), (7.5f), (7.6a) - (7.6d) \tag{7.7b}$$

$$z_{i1}^{N_i} = t_{i1}^{n_{i1}}, \forall N_i \in \mathbb{N}_i, i \in S \tag{7.7c}$$

$$L_i^{\mathrm{E}} \geq \sum_{N_i \in \mathbb{N}_i} \left[\Omega(N_i) z_{i|C|}^{N_i} \right], \forall i \in S. \tag{7.7d}$$

7.1.4 Minimize worst-case expected propagation latency

Sometimes, the worst expected propagation latency among all switches can be a key metric for the network performance. The worst-case expected propagation latency among $L_i^{\mathrm{E}}, i \in S$, which is denoted by W, is given by,

$$W = \max_{i \in S} L_i^{\mathrm{E}}. \tag{7.8}$$

The second assignment problem is considered to find an assignment of controllers to switches that minimizes the worst-case expected propagation latency (MinWEL), which is expressed as the following MILP problem.

$$\min \quad W \tag{7.9a}$$

$$\text{s.t.} \quad (7.4c) - (7.4f), (7.5c), (7.5d), (7.5f), (7.6a) - (7.6d),$$
$$(7.7c) \tag{7.9b}$$

$$W \geq \sum_{N_i \in \mathbb{N}_i} \left[\Omega(N_i) z_{i|C|}^{N_i} \right], \forall i \in S. \tag{7.9c}$$

7.1.5 Maximize expected number of switches within propagation latency bound

Let b_i denote a propagation latency bound for switch $i \in S$. The probability that switch $i \in S$ is within the propagation latency bound b_i, which is represented by P_i, is obtained by,

$$P_i = \sum_{j \in C: x_{ij}=1, l_{ij} \leq b_i} \prod_{k \in C: x_{ik}=1, y_{ik} < y_{ij}} p_k(1 - p_j). \tag{7.10}$$

Equation (7.10) sums the possibilities of all the cases where the propagation latency between switch $i \in S$ and its master controller is not greater than b_i. Therefore, the expected number of switches within a propagation latency bound, J, is expressed by,

$$J = \sum_{i \in S} P_i. \tag{7.11}$$

Given $N_i \in \mathbb{N}_i$, we obtain the probability that switch $i \in S$ is within the propagation latency bound b_i by using (7.10) that is expressed as $P_i = \Gamma(N_i)$. By using the similar methods obtaining (7.7a)-(7.7d), this work expresses the master and slave controller assignment to maximize the expected number of switches within a propagation latency bound (MaxENS) problem as following MILP problem.

$$\max \quad \sum_{i \in S} P_i \tag{7.12a}$$

$$\text{s.t.} \quad (7.4c) - (7.4f), (7.5c), (7.5d), (7.5f), (7.6a) - (7.6d),$$

$$(7.7c) \tag{7.12b}$$

$$P_i \leq \sum_{N_i \in \mathbb{N}_i} \left[\Gamma(N_i) z_{i|C|}^{N_i} \right], \forall i \in S. \tag{7.12c}$$

7.2 Analysis for priority policy

Given an assignment of $C_i = \{j | j \in C : x_{ij} = 1\}$ for switch $i \in S$, let $d \in D$ denote a possible permutation for the controllers in C_i, where D with the size of $|D| = |C_i|!$ represents the set of all possible permutations. Each permutation in

$\boldsymbol{D}$ represents a priority setting for y_{ij}, where a controller with the lower order has a higher priority to become as the master controller. This work provides the following theorems to analyze the optimality of adopting the priority policy of LLFP.

Theorem 10 *The permutation satisfying that a controller with lower latency is with a smaller order, or adopting LLFP, results in the minimum value of* L_i^{E} *among all permutations in* $\boldsymbol{D}$.

Proof : This work first proves a necessary condition to maximize the value of L_i^{E}. Let d^* denote the permutation achieving the minimum value of L_i^{E}, or $d^* = \arg\min_{d \in D} L_i^{\mathrm{E}}$. Suppose that there exists a pair of two controllers of j, $j' \in C_i$ with $l_{ij} < l_{ij'}$ and with the adjacent orders, where the order of controller j is larger by one than that of controller j', in permutation d^*. Let d' represent a permutation, where the orders of controllers j and j' are switched and the orders of other controllers are kept as the same with those in d^*.

According to (7.2), the values of L_i^{E} with d^* and d' are computed as $L_i^{\mathrm{E}*} = a+b[(1-p_{j'})l_{ij'}+p_{j'}(1-p_j)l_{ij}]+c$ and $L_i^{\mathrm{E}\prime} = a+b[(1-p_j)l_{ij}+p_j(1-p_{j'})l_{ij'}]+c$, respectively, where $L_i^{\mathrm{E}*}$ and $L_i^{\mathrm{E}\prime}$ have the same parts of a, b, and c, which are related to other controllers. For example, b is the product of failure probabilities of controllers whose orders are smaller than those of controllers j and j'. Therefore, $L_i^{\mathrm{E}*} - L_i^{\mathrm{E}\prime} = b[(1-p_{j'})l_{ij'} + p_{j'}(1-p_j)l_{ij} - (1-p_j)l_{ij} - p_j(1-p_{j'})l_{ij'}] = b(1-p_{j'})(1-p_j)(l_{ij'} - l_{ij}) > 0$. It contradicts that $d^* = \arg\min_{d \in D} L_i^{\mathrm{E}}$. Hence, for any pair of two adjacent controllers in permutation d^*, the one having a lower latency is with a smaller order than the other. It indicates a necessary condition that the controllers are sorted as a non-decreasing order of latency, or with adopting LLFP, in permutation d^*.

Then this work shows that the necessary condition is also a sufficient condition to minimize the value of L_i^{E}. Since some controllers may have the same latency, there can be multiple permutations satisfying this necessary condition, but all of them result in the same value of L_i^{E} based on the above analyses. Therefore, a permutation with adopting LLFP leads to the minimum value of L_i^{E}. Theorem 10 is proved. □

Theorem 11 *LLFP achieves the optimal objectives in the problems of MinAEL and MinWEL compared to other policies.*

Proof : For a given priority policy, there is a permutation in D corresponding to it. Based on Theorem 10, considering the objectives of MinAEL and MinWEL, we easily obtain that LLFP achieves the optimal objectives in MinAEL and MinWEL compared to other policies. □

Theorem 12 *The permutation satisfying that a controller with the latency not greater than the latency bound has a smaller order than another controller with the latency greater than the latency bound, or with adopting LLFP, achieves the maximum value of P_i among all permutations in D.*

Proof : Similar to the proof for Theorem 10, this work first proves a necessary condition. Now, let d^* represent the permutation achieving the maximum value of P_i, or $d^* = \arg\max_{d \in D} P_i$. Suppose that there exists a pair of two adjacent controllers of j, $j' \in C_i$,where $l_{ij} \leq b_i$, $l_{ij'} > b_i$, and the order of controller j is larger by one than that of controller j', in permutation d^*. Let d' represent a permutation, where the orders of controllers j and j' are switched and the orders of other controllers are kept as the same with those in d^*.

According to (7.10), the values of P_i with d^* and d' are computed as $P_i^* = a+bp_{j'}(1-p_j)+c$ and $P_i' = a+b(1-p_j)+c$, respectively, where P_i^* and P_i' have the same parts of a, b, and c; b is the product of failure probabilities of controllers whose orders are smaller than those of controllers j and j'. Therefore, $P_i^* - P_i' = b(p_{j'} - 1)(1 - p_j) < 0$. It contradicts that $d^* = \arg\max_{d \in D} P_i$. Hence, we obtain a necessary condition that, for any pair of two adjacent controllers in permutation d^*, if the latency of one controller is not greater than the latency bound and that of the other is greater than the latency bound, the one with smaller latency has a smaller order than the other. In other words, for a controller with the latency not greater than the latency bound, all controllers having smaller orders must be with the latency not greater than the latency bound.

Then this work shows that the necessary condition is also a sufficient condition. For a permutation satisfying the necessary condition, switch i is within the latency bound, if and only if there is at least one controller that is with the latency not greater than the latency bound and does not fail. We obtain

that $P_i = 1 - \prod_{j \in C_i : l_{ij} \leq b_i} p_j$, where $\prod_{j \in C_i : l_{ij} \leq b_i} p_j$ is the probability that all controllers with the latency not greater than the latency bound fail, which is fixed for given C_i. It indicates that, regardless of the exact order, all permutations satisfying the necessary condition have the same value of $P_i = 1 - \prod_{j \in C_i : l_{ij} \leq b_i} p_j$, which is the maximum value.

Clearly, a permutation with adopting LLFP satisfies the necessary and sufficient condition. Theorem 12 is proved. □

Theorem 13 *LLFP achieves the optimal objective in the problem of MaxENS compared to other policies.*

Proof : Based on Theorem 12, considering the objective of MaxENS, we easily obtain that LLFP achieves the optimal objective in MaxENS compared to other policies. □

7.3 NP-completeness

This work shows that all the three problems introduced in Section 7.1 are NP-complete. The proof for the MaxENS problem is described below, which is similar to the proofs for other two problems.

This work first considers a MaxENS without the survivability guarantee (MaxENS-NSG) problem, where the constraint of survivability guarantee in (7.1) is released from MaxENS. This work defines the decision version of MaxENS-NSG problem as below:

Problem Given a set of switches of S and a set of controllers of C, is it possible to find an assignment of controllers to switches so that the expected number of switches within the propagation latency bound is no less than l?

Lemma 2 *If two positive variables of a and b satisfy $a \times b = c$, where c is a constant, the value of $a + b$ achieves the minimum value of $2\sqrt{c}$ when $a = b = \sqrt{c}$.*

Proof : It can be easily proved considering the properties of function of $f(a) = a + c/a$. □

Theorem 14 *The MaxENS-NSG decision problem is NP-complete.*

Proof : If a certificate of any instance of the MaxENS-NSG decision problem is given, we need to compute (7.12a) for verification. Based on the proof for Theorem 12, the value of (7.12a) can be computed by $\sum_{i \in S}(1 - \prod_{j \in C_i : l_{ij} \leq b_i} p_j)$. Therefore, for each switch, we check whether the latency between an assigned controller and the switch is within the bound; if it is, we update the value of (7.12a) by incorporating the failure probability of controller, which takes $O(|S||C|)$ in total. As a result, we can verify whether a certificate of any instance of the MaxENS-NSG decision problem has the expected number of switches within the propagation latency bound no less than l in a polynomial time of $O(|S||C|)$. Therefore, the MaxENS-NSG decision problem is NP.

Then, this work shows that PP, which is a known NP-complete problem [90], is reducible to the MaxENS-NSG decision problem.

This work constructs an instance of the MaxENS-NSG decision problem from any instance of PP. An instance of PP consists of a multiset of positive integers of G and the value of positive integer $g \in G$ is represented by I_g. An instance of the MaxENS-NSG decision problem is constructed with the following algorithm, which performs in a polynomial time of $O(|G|)$.

1) Set $|S| = 2$.

2) Set $|C| = |G|$. For each positive integer $g \in G$, there is a corresponding controller $j \in C$ with the failure possibility of $p_j = \mathrm{e}^{-I_g}$.

3) For each controller $j \in C$, set $c_j = 1$, which means that each controller can control at most one switch.

4) Set $l_{ij} \leq b_i$, $i \in S$, $j \in C$, or the latency between any pair of switch and controller satisfies the latency bound.

5) Set $l = 2 - 2\mathrm{e}^{\frac{-\sum_{g \in G} I_g}{2}} = 2 - 2\sqrt{\prod_{j \in C} p_j}$.

Consider that a PP instance is a Yes instance. G is able to be partitioned into two subsets of G_1 and G_2, and the sum of numbers in each subset is $\frac{\sum_{g \in G} I_g}{2}$. Define a MaxENS-NSG instance from the PP instance by using the above described algorithm. In the MaxENS-NSG instance, the set of controllers of

C is able to be partitioned into two subsets C_1 and C_2, which refer to G_1 and G_2, respectively, and the product of failure probabilities of controllers in each subset is $e^{\frac{-\sum_{g\in G} l_g}{2}}$. By assigning C_1 and C_2 to the two switches, respectively, we obtain the expected number of switches within the propagation latency bound is $2-2e^{\frac{-\sum_{g\in G} l_g}{2}}$, which is not less than l. As a result, it is possible to find an assignment of controllers to switches so that the expected number of switches within the propagation latency bound is no less than l. Therefore, the MaxENS-NSG instance is a Yes instance.

Conversely, consider that a MaxENS-NSG instance is a Yes instance. In the Yes MaxENS-NSG instance, it is possible to find the assignment of C_1 and C_2 making the value of (7.12a) no less than l. Based on Lemma 2, the maximum value of (7.12a), which can be expressed by $2-(\prod_{j\in C_1} p_j + \prod_{j\in C_2} p_j)$, is $l = 2-2\sqrt{\prod_{j\in C} p_j}$. It indicates that $\prod_{j\in C_1} p_j = \prod_{j\in C_2} p_j = e^{\frac{-\sum_{g\in G} l_g}{2}}$. Therefore, by referring to C_1 and C_2, we are able to partition G into two subsets G_1 and G_2 such that the two sums of numbers in the two subsets equal to each other. Therefore, if the MaxENS-NSG instance is a Yes instance, then the PP instance is a Yes instance.

Note that the above described algorithm transforms any PP instance into a MaxENS-NSG instance in a polynomial time. This work confirmed that if a PP instance is a Yes instance, then the corresponding MaxENS-NSG instance is a Yes instance, and vice versa. This proves that PP, a known NP-complete problem, is polynomial time reducible to the MaxENS-NSG decision problem. Thus, the MaxENS-NSG decision problem is NP-complete. □

Theorem 15 *The MaxENS problem is NP-complete.*

Proof : Given a certificate of any instance of MaxENS, we need compute (7.4a) in addition to (7.12a) for verification, which requires $O(|S||C|)$. Therefore, the MaxENS problem is NP.

Clearly, the MaxENS-NSG problem is a subset of the MaxENS problem by setting $q_i = 1, \forall i \in S$. Based on Theorem 14, the MaxENS problem is NP-complete. □

By adopting the methods similar with the above proofs, we easily obtain that both problems described in Sections 7.1.3 and 7.1.4 are NP-complete.

7.4 Heuristic algorithm

A controller can be assigned to multiple switches, and a switch can have multiple controllers in the proposed model. It can be viewed as an extension of classical WBM model. This section begins by introducing the WBM problem, and then presents a polynomial-time greedy algorithm to solve the problems described in Section 7.1.

7.4.1 Weighted bipartite b-matching problem

Consider an undirected bipartite graph $G(V, E)$, where V is a set of nodes and E is a set of edges. E consists of all the edges between two nodes that are on different sides of G and are allowed to be connected. The WBM problem, where the nodes on one side of G can be matched to several nodes on the other side, is widely studied in computer science and economics, such as thesiss-reviewers matching [118] and recommendation systems [119, 120]. An instance of matching the nodes on one side to the nodes on the other side is represented by a subgraph of $G'(V, E')$, where $E' \subseteq E$. Usually, the goal of WBM problem is to find a subgraph G' such that the total weight of the matched edges in E' is maximized or minimized with satisfying all constraints. The term of b-matching means that the number of matched edges in E' for node $v \in V$ is required to be in a range of $[b_v^{\mathrm{L}}, b_v^{\mathrm{U}}]$ or $[b_v^{\mathrm{L}}, \infty)$, where b_v^{L} and b_v^{U} are always given as two constants for node v.

In the proposed model, $V = S \cup C$; $E = \{(i, j) | i \in S, j \in C : (i, j) \notin H\}$ consists of all the edges of (i, j), where switch i is allowed to be connected with controller j, or $(i, j) \notin H$. The weight of each edge in E represents the propagation latency between a switch in S and a controller in C. The matched edges in $E' = \{(i, j) | i \in S, j \in C_i\}$ are determined by the master and slave controller assignment. For each controller $j \in C$, the number of matched edges is required to be in a range of $[0, c_j]$, which is the same with its capacity constraint.

In addition to the above similarity, there are two main differences between the considered problems and the classical WBM problem. Firstly, the objective in each considered problem is different with that of the WBM problem. It has been known that the objective with minimizing or maximizing the total

weight of matched edges in the WBM problem is a linear and modular function [86, 121]; the properties of objectives in the considered problems need to be analyzed. Secondly, for each switch $i \in S$, the number of matched edges is required to be in a range of $[b_i^{\mathrm{L}}, \infty)$, but different from other WBM problems, such as those described in [118–120], b_i^{L} here is not a constant for switch i; it depends on the survivability guarantee for switch i and the failure probabilities of assigned controllers. While the WBM problem can be solved efficiently by polynomial-time algorithms, such as the one presented in [122], all the three considered problems are NP-complete; even a subproblem without the survivability guarantee, i.e., MaxENS-NSG, is NP-complete.

7.4.2 Lower-bound aware greedy weighted bipartite *b*-matching algorithm

This work introduces a lower-bound aware greedy weighted bipartite b-matching (LA-GWBM) algorithm (see Algorithm 12) that incrementally satisfies the survivability guarantees, or the lower bound of b_i^{L}, for all switches. Let U denote a set of switches, in each of which the unavailability does not satisfy the survivability guarantee. Let T denote a set of controllers, each of which has remaining capacity to hold at least one more switch. $M \subseteq E$ represents a set of edges which can be selected. Therefore, $U = S$, $T = C$, $M = E$, and $E' = \emptyset$ at the beginning of algorithm. The marginal gain in a corresponding objective function by adding edge $(i, j) \in M$ to E' is defined as $\Delta[E', (i, j)] = f[E' \cup \{(i, j)\}] - f(E')$, where f represents the corresponding objective function, such as (7.3), (7.8), and (7.11). LA-GWBM gives preference to the switches that do not satisfy the survivability guarantees. In each iteration, LA-GWBM assigns controller $j \in T$ to switch $i \in U$, where matched edge $(i, j) \in M$ leads to an optimal value for the marginal gain of $\Delta[E', (i, j)]$ in the corresponding objective function. For example, when LA-GWBM is applied to solve the MaxENS problem, in each iteration, this work selects edge $(i, j) \in M$ by maximizing the value of $\Delta[E', (i, j)]$ with considering the objective function of (7.11). After the assignment, the selected edge (i, j) is deleted from M; switch i is deleted from U if its survivability guarantee is satisfied, or $\sum_{j \in C_i} -\log p_j \geq -\log q_i$; controller j is deleted from T and all edges connecting controller j are deleted from M if

the capacity of controller j is used up, or $c_j = 0$.

For MinAEL and MinWEL, as long as the survivability guarantee of a switch is satisfied, no more additional controller is assigned to the switch to achieve the objective value as low as possible. Therefore, LA-GWBM for MinAEL and MinWEL terminates, if any one of U, T, and M becomes an empty set. For MaxENS, as long as there exists a controller having remaining capacity, the controller is assigned to switches that have not been matched to it to obtain the objective value as great as possible. When U becomes empty, LA-GWBM for MaxENS keeps assigning controller $j \in T$ to switch $i \in S$ with the same greedy manner, where $(i, j) \in M$, until any of T and M becomes empty.

7.4.3 Computational time complexity

The LA-GWBM algorithm sorts controllers for each switch to obtain y_{ij}, $i \in S$, $j \in C$, at the beginning of algorithm. The computational time complexity of sorting $|C|$ controllers for $|S|$ times is $O(|S||C| \log |C|)$. For a candidate edge of $(i, j) \in E$, the values of $\Delta[E', (i, j)]$ in terms of (7.3) or (7.8), and (7.11) are obtained by computing (7.2), and (7.10), respectively. Through reusing the past computation results, both computational time complexities of computing (7.2) and (7.10) are $O(|C|)$. In each iteration, the LA-GWBM algorithm computes (7.2) or (7.10) at most $|S||C|$ times to select an optimal edge. Equation (7.1) is also computed in each iteration, where the computational time complexity is reduced from $O(|C|)$ to $O(1)$ by reusing the past computation results. Hence, the computational time complexity of each iteration in the LA-GWBM algorithm is $O(|S||C|^2)$. For each switch, the number of required iterations is at most $|C|$, which leads to at most $|S||C|$ iterations in the LA-GWBM algorithm. Therefore, the computational time complexity of LA-GWBM algorithm is $O(|S|^2|C|^3)$.

7.4.4 Approximation performance

This work analyzes the approximation performance of LA-GWBM algorithm, when it is used to solve the MaxENS problem. The analysis is related to

the concepts of submodular function, p-system, and matroid; their detailed definitions can be found in literatures, such as [86].

Let $\Theta(E')$ denote the objective function of MaxENS. Based on Theorem 12, given an assignment of $E' \in E$, the expected number of switches within the latency bound is computed by $\Theta(E') = \sum_{i \in S}(1 - \prod_{j \in C:(i,j) \in E', l_{ij} \leq b_i} p_j)$.

Theorem 16 *The objective function of MaxENS, $\Theta(E')$, is a monotone submodular function.*

Proof : Firstly, it is easy to obtain that, for any $E' \subseteq E$ and $(i, j) \in E \setminus E'$, $\Delta[E', (i, j)] = \Theta[E' \cup \{(i, j)\}] - \Theta(E') \geq 0$. Hence, $\Theta(E')$ is a monotone function.

Then, consider two arbitrary subsets, E', $E^* \subseteq E$, where $E' = E^A \cup E^B$ and $E^* = E^B \cup E^C$; E^A, E^B, and E^C are three disjoint sets. For switch $i \in S$ and E^A, this work defines $P_i^A = \prod_{j \in C:(i,j) \in E^A, l_{ij} \leq b_i} p_j$; the same definition is applied to E^B and E^C. We have,

$$\Theta(E') + \Theta(E^*) = \sum_{i \in S}(1 - P_i^A P_i^B) + \sum_{i \in S}(1 - P_i^B P_i^C), \tag{7.13}$$

$$\Theta(E' \cup E^*) + \Theta(E' \cap E^*) = \sum_{i \in S}(1 - P_i^A P_i^B P_i^C) + \sum_{i \in S}(1 - P_i^B). \tag{7.14}$$

Therefore, we obtain,

$$\Theta(E') + \Theta(E^*) - \Theta(E' \cup E^*) - \Theta(E' \cap E^*) = \sum_{i \in S} P_i^B (1 - P_i^A)(1 - P_i^C). \tag{7.15}$$

Since P_i^A, P_i^B, and P_i^C are in the range of [0, 1], (7.15) is with a non-negative value. Hence, $\Theta(E')$ is a submodular function.

Therefore, the objective function of MaxENS, $\Theta(E')$, is a monotone submodular function. □

Consider a special case of $q_i = 1, \forall i \in S$, i.e., MaxENS-NSG. Then the lower bound of b_i^{L} for each switch becomes 0; only the upper bound of capacity constraint for each controller is considered. Several works, such as [86, 123], show that a b-matching constraint, where each node v has a finite capacity, or is associated with a range of $[0, b_v^{\mathrm{U}}]$, is a 2-system constraint. For the constraint of MaxENS-NSG, where each controller has a range of $[0, b_j^{\mathrm{U}}]$, but each switch has a range of $[0, \infty)$, this work provides the following theorem.

Theorem 17 *The constraint of MaxENS-NSG is a 1-system, or matroid, constraint.*

Proof : Let F_{NSG} denote the family of feasible solutions for MaxENS-NSG. Consider E', $E^* \subseteq E$ as two solutions. This work will show that the pair of (E, F_{NSG}) forms a matroid.

First, this work show that the hereditary property of matroid is satisfied. Clearly, an empty set is a feasible solution satisfying the cardinality constraint of MaxENS-NSG. If E^* is a feasible solution, or $E^* \in F_{\mathrm{NSG}}$, and $E' \subseteq E^*$, E' must be a feasible solution, or $E' \in F_{\mathrm{NSG}}$.

Then, this work shows that the exchange property of matroid is satisfied. If E' and E^* are feasible solutions and $|E^*| > |E'|$, there exists at least one controller of j that is with the number of connected switches in E^* at least larger by one than that in E'; let i denote one of the additional connected switches in E^* for controller j. Since each switch has an infinite capacity, adding edge (i, j) to E' leads to another feasible solution, or $\exists (i, j) \in E^* \setminus E'$ such that $E' \cup \{(i, j)\} \in F_{\mathrm{NSG}}$.

Therefore, (E, F_{NSG}) is a matroid; Theorem 17 is proved. □

Theorem 18 *For the special case of $q_i = 1, \forall i \in S$, LA-GWBM achieves a 1/2-approximation of the optimal objective value.*

Proof : The work in [86] proves that a natural greedy algorithm maximizing a monotone submodular objective function achieves $1/(p+1)$-approximation for a p-system constraint. Therefore, based on Theorems 16 and 17, LA-GWBM achieves 1/2-approximation of the optimal value for the special case. □

When the survivability guarantee is considered, i.e., switch $i \in S$ has a positive (non-constant) lower bound of b_i^{L}, the family consisting of feasible solutions of MaxENS is not an independence family [86], i.e., the hereditary property of matroid is not satisfied. For this general case, the theoretical analysis for the approximation guarantee of LA-GWBM becomes difficult. Section 7.5 will show that LA-GWBM performs much better than the lower bound of 1/2-approximation.

7.5 Numerical results

In the numerical analysis, this work shows the optimal assignments for different problems introduced in Section 7.1 with considering the same network. This work considers an approach usually adopted in literature [33, 35] as a baseline, where the order to connect controllers is determined by introducing some decision variables instead of by adopting a policy-based approach. This work compares the results from the proposed model and those from the baseline in terms of the objective values and the computation time. For the proposed model, this work also compares the results obtained by solving the MILP problem with those by running the introduced greedy algorithm. The performance dependencies on the numbers of switches and controllers, the capacity and failure probability of each controller, and the acceptable unavailability of each switch are evaluated.

This work uses Intel Core i7-7700 3.60 GHz 4-core CPU with 32 GB memory to solve the MILP problems or run the heuristic algorithm. The MILP problems are solved by the IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer with the version of 12.8 [88].

7.5.1 Optimal assignments for different problems

This work considers a network with the given information shown in Fig. 7.1(a) , where a switch is allowed to be connected with any controller, i.e., $\boldsymbol{H} = \emptyset$.

Table 7.2 shows the assigned controllers for each switch in optimal assignments for the MinAEL problem, the MinWEL problem, the MaxENS problem with considering propagation latency bound as 10 μs, and the MaxENS problem with considering propagation latency bound as 10^4 μs, respectively. This work observes that the optimal assignments vary for different problems. The difference between the optimal assignments of MinAEL problem and that of MinWEL problem is the assigned controllers for switches 4 and 5, where controller 1 can only be assigned to one of switches 4 and 5 due to its capacity constraint. When controller 1 is assigned to any one of them, controllers 2 and 3 have to be assigned to the other to satisfy the survivability guarantee, which leads to two options. For the option that controller 1 is assigned to switch 5, the expected propagation latencies of switches 4 and 5 are 10800 μs

and 9.99 μs, respectively. For the option that controller 1 is assigned to switch 4, the expected propagation latencies of switches 4 and 5 are 9990 μs and 1890 μs, respectively. As a result, the former option is chosen in the MinAEL problem to obtain a less average-case propagation latency as 5404.995 μs, and the latter option is chosen in the MinWEL problem to obtain a less worst-case propagation latency as 9990 μs.

Table 7.2: Assigned controllers in optimal assignments for different problems.

Switch	Assigned controllers			
i	MinAEL	MinWEL	MaxENS 10 μs	MaxENS 10^4 μs
1	1, 3	1, 3	1, 3	1, 2, 3
2	1, 3	1, 3	1, 3	1, 3
3	1	1	2, 3	1, 3
4	2, 3	1	2, 3	1, 2
5	1	2, 3	1	2, 3
6	2, 3	2, 3	1, 2	2, 3

For the optimal assignments of MaxENS problem, this work observes that controller 1 is assigned to switches 5 and 6, and switches 3 and 4, respectively, when the propagation latency bounds are considered as 10 μs and 10^4 μs. This is because the propagation latency between controller 1 and switch 3 or 4 exceeds 10 μs and that between controller 1 and switch 5 or 6 does not. Note that controller 1 can be assigned to only two of switches 3, 4, 5, and 6 due to the capacity constraint. As a result, controller 1 is assigned to switches 5 and 6 when the propagation latency bounds are considered as 10 μs. All the propagation latencies between controller 1 and switches 3, 4, 5, and 6 are within 10^4 μs, and assigning controller 1 to switches 3 and 4 increases the objective value more than assigning it to any other two switches. Therefore, controller 1 is assigned to switches 3 and 4 when the propagation latency bound is considered as 10^4 μs.

Furthermore, this work observes that all the controllers are assigned to switch 1 only for the MaxENS problem with 10^4 μs propagation latency bound. Assigning controllers 1 and 3 to switch 1 satisfies its survivability guarantee.

Further adding the assignment of controller 2 to switch 1, or replacing controller 3 with controller 2, increases the objective values for the MinAEL problem and the MinWEL problem, and does not affect the objective value for the MaxENS problem with 10 μs propagation latency bound. In other words, these three problems achieve the optimal objective values by assigning controllers 1 and 3 to switch 1. However, the objective value for the MaxENS problem, which is a maximization problem, with 10^4 μs propagation latency bound is increased by further assigning controller 2 to switch 1. As a result, all the controllers are assigned to switch 1 to obtain the optimal objective value for the MaxENS problem with 10^4 μs propagation latency bound.

7.5.2 Competitive evaluation

Baseline

Let D with $|D| = |C|!$ represent the set of all permutations for all controllers. A permutation, or order, in D corresponds to one priority setting of y_{ij} and vise versa. For the baseline, y_{ij} is not given information based on LLFP, but is determined by the selection of permutation from D. Let w_{id} denote a binary decision variable; $w_{id} = 1$ if permutation $d \in D$ is selected for switch $i \in S$ and zero otherwise. Given the assignment of $N_i \in \mathbb{N}_i$ and the order of d, the values of L_i^{E} and P_i are computed by (7.2) and (7.10), which are represented by the functions of $\Omega(N_i, d)$ and $\Gamma(N_i, d)$, respectively. Let $v_i^{N_i d} = z_{i|C|}^{N_i} w_{id}$ denote a binary variable introduced for linearization; $v_i^{N_i d}$ is set to one if the assignment of N_i and the order of d are selected and zero otherwise. The MinAEL problem with considering the baseline approach is formulated as the following MILP problem.

$$\min \quad \frac{1}{|S|} \sum_{i \in S} L_i^{\mathrm{E}} \tag{7.16a}$$

$$\text{s.t.} \quad (7.4c) - (7.4f), (7.5c), (7.5d), (7.5f), (7.6a) - (7.6d), (7.7c) \tag{7.16b}$$

$$\sum_{d \in D} w_{id} = 1, \forall i \in S \tag{7.16c}$$

$$v_i^{N_i d} \leq z_{i|C|}^{N_i}, \forall i \in S, N_i \in \mathbb{N}_i, d \in D \tag{7.16d}$$

$$v_i^{N_i d} \leq w_{id}, \forall i \in S, N_i \in \mathbb{N}_i, d \in D \tag{7.16e}$$

$$v_i^{N_i d} \geq z_{i|C|}^{N_i} + w_{id} - 1, \forall i \in S, N_i \in \mathbb{N}_i, d \in D \tag{7.16f}$$

$$L_i^{\mathrm{E}} \geq \sum_{d \in D} \sum_{N_i \in \mathbb{N}_i} \left[\Omega(N_i, d) v_i^{N_i d} \right], \forall i \in S \tag{7.16g}$$

$$w_{id} \in \{0, 1\}, \forall i \in S, d \in D \tag{7.16h}$$

$$v_i^{N_i d} \in \{0, 1\}, \forall i \in S, N_i \in \mathbb{N}_i, d \in D. \tag{7.16i}$$

Equation (7.16c) ensures that an order to connect the controllers is set for each switch. Equations (7.16d)-(7.16f) linearize the product of $z_{i|C|}^{N_i} w_{id}$. Equation (7.16g) computes the value of L_i^{E} for given controller assignment and connecting order. By following the similar methods, we can formulate the MILP problems of MinWEL and MaxENS for the baseline.

Experiment setup

This work conducts 500 trials to compute the average values of objective values and computation times. In each trial, the capacity of each controller, the failure probability of each controller, the acceptable unavailability of each switch, and the propagation latency between a switch and a controller, are randomly selected from the ranges of [5, 20], $[10^{-4}, 10^{-1}]$, $[10^{-5}, 10^{-1}]$ and $[10, 10^5]$ μs, respectively. For MaxENS, the latency bound for each switch is randomly set within the range of $[10, 10^5]$ μs. Let η denote a ratio of the expected number of switches within the bound to the total number of switches.

With unlimited admissible computation time

This work considers the unlimited admissible computation time to obtain the optimal values by solving the MILP problems. The number of controllers is set to four; the performance dependency on the number of switches is evaluated. Note that, since the computation time of baseline to obtain the optimal solution for one trial is more than one day when $|S| \geq 18$, the results of baseline are only shown for $|S| < 18$.

Figure 7.2 presents the average-case expected propagation latency obtained by the proposed model with different approaches and the baseline for different numbers of switches. This work observes that the proposed model obtains the same objective values as the baseline for MinAEL, which is held with

Theorem 11. As the number of switches increases, the ratio of objective value obtained by LA-GWBM to the optimal value increases; more specifically, it increases from 1.00 to 1.06. This is because the number of feasible solutions increases as the number of switches increases, which degrades the efficiency of introduced greedy algorithm. In addition, since this work considers the average-case expected propagation latency, the optimal values with different numbers of switches are comparable.

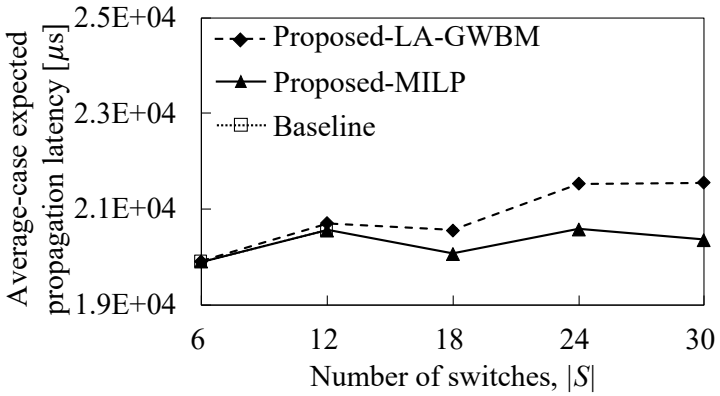


Figure 7.2: Comparison among average-case expected propagation latency obtained by proposed model with different approaches and baseline.

Figure 7.3 compares the worst-case expected propagation latency obtained by the proposed model with different approaches and the baseline for different numbers of switches. It shows that the same objective values are obtained by the proposed model and the baseline for MinWEL, which is kept with Theorem 11. The ratio of objective value obtained by LA-GWBM to the optimal value increases from 1.00 to 1.14 as the value of $|S|$ increases. Introducing more switches increases the probability to involve some relatively large propagation latency. As a result, the worst-case expected propagation latency increases as the number of switches increases.

Figure 7.4 presents the values of η obtained by the proposed model with different approaches and the baseline for different numbers of switches; the lower bound of Theorem 18 is also shown. This work observes that the numerical result follows Theorem 13, where LLFP achieves the optimal value for

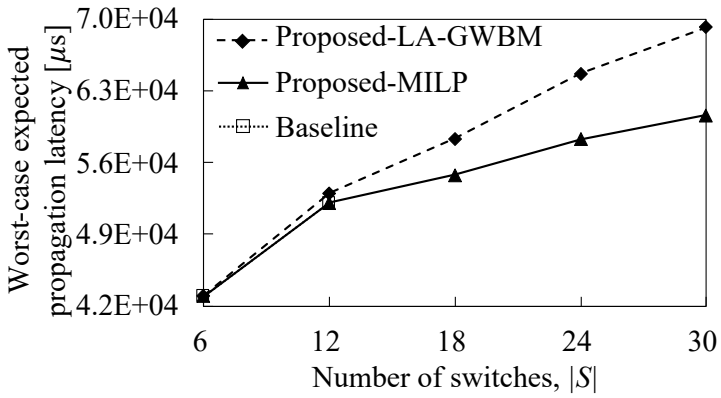


Figure 7.3: Comparison among worst-case expected propagation latency obtained by proposed model with different approaches and baseline.

MaxENS compared to other policies. The objective value obtained by LA-GWBM is in average 0.99 times less than the optimal value, which performs better than the lower bound of Theorem 18 as shown in Fig. 7.4.

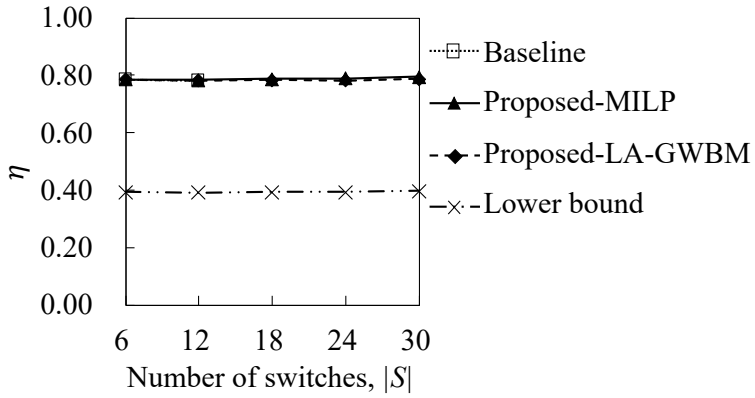


Figure 7.4: Comparison among values of η obtained by proposed model with different approaches and baseline.

Table 7.3 shows the computation times to obtain the results of Fig. 7.4. For each model or approach, the computation time increases as the number of switches increases. Benefiting from the theorems presented in Section 7.2, the computation time of proposed model to obtain the optimal solution by solving

the MILP problem is about 10^2 times less than that of baseline when $|S| < 18$; the computation time of proposed model is within 20 [s] when $18 \leq |S| \leq 30$, but this work cannot obtain the optimal solution of baseline for one trial within one day ($= 8.64 \times 10^4$ [s]). This work observes that, for the proposed model, the heuristic of LA-GWBM requires the computation time about 10^3 times less than that required by solving the MILP problem.

Table 7.3: Computation times [s] to obtian results of Fig. 7.4.

$\|S\|$	Baseline	Proposed-MILP	Proposed-LA-GWBM
6	53.13	1.90×10^{-1}	6.50×10^{-4}
12	1.14×10^2	5.51	2.17×10^{-3}
18	$\geq$ one day	7.61	3.68×10^{-3}
24	$\geq$ one day	13.55	5.34×10^{-3}
30	$\geq$ one day	18.82	7.43×10^{-3}

With limited admissible computation time

When the number of controllers increases, this work observes that the computation times to obtain the optimal solutions by solving the MILP problems increase greatly. Let T denote the admissible computation time. This work compares the results obtained by different models and approaches within T [s], where $|S|$ is set to 12. Note that, for the baseline, there is no feasible solution returned when $|C| \geq 7$ and $T = 60$.

Figure 7.5 presents the average-case expected propagation latency obtained by different models and approaches with different values of admissible computation time for different numbers of controllers. When the problem size is small, or $|C| = 4$, the proposed model and the baseline obtain the minimized objective value within 5 and 60 [s], respectively, by solving the MILP problems. When $|C| \geq 5$, the baseline with $T = 60$ always provides the largest objective value. The objective values of proposed model by solving the MILP problem with $T = 5$, 30, and 60 are larger than those obtained by running LA-GWBM when $|C| \geq 5$, 6, and 6, respectively. It indicates that, when the value of $|C|$ is small or the admissible computation time is large, solving the MILP problem for the proposed model can achieve a smaller objective value; otherwise, running the LA-GWBM heuristic can provide a better solution. This

work observes that the objective value obtained by each approach for the proposed model decreases as the number of controllers increases. This is because more controllers can provide more choices with lower propagation latency for switches. However, for the baseline with $T = 60$, the objective value increases as the number of controllers increases. This is because the number of feasible solutions searched within the same computation time greatly decreases as the problem size increases for the baseline.

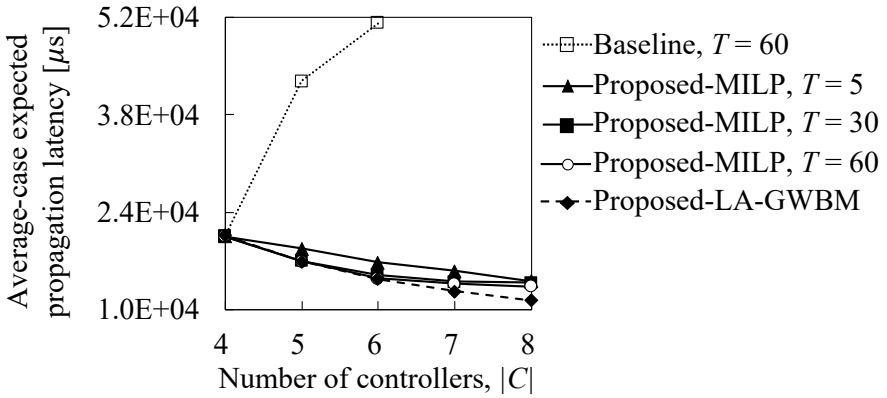


Figure 7.5: Comparison among average-case expected propagation latency obtained by different models and approaches with limited admissible computation time.

Figure 7.6 compares the worst-case expected propagation latency obtained by different models and approaches with different values of T for different numbers of controllers. This work obtains the similar observations to those obtained from Fig. 7.5.

Figure 7.7 shows the values of η obtained by different models and approaches with different values of T for different numbers of controllers. As the number of controllers increases, each switch can connect more controllers with the propagation latency within the bound. Consequently, the objective value obtained by each approach for the proposed model increases as the value of $|C|$ increases. However, since the number of searched feasible solutions decreases as the problem size increases, the objective value obtained by the baseline with $T = 60$ decreases.

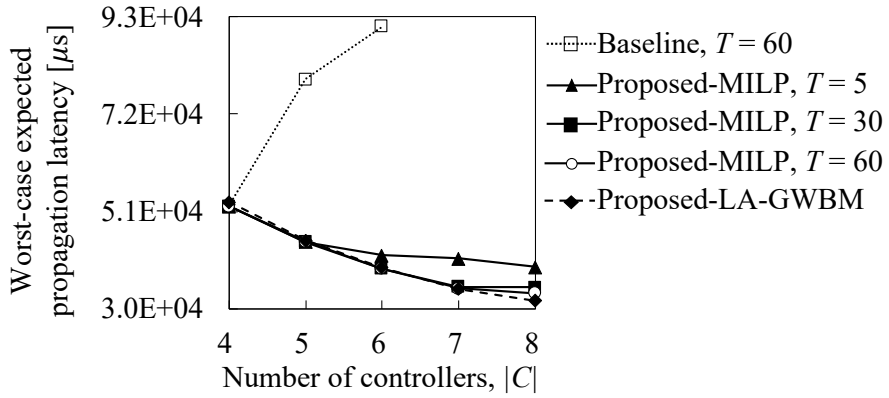


Figure 7.6: Comparison among worst-case expected propagation latency obtained by different models and approaches with limited admissible computation time.

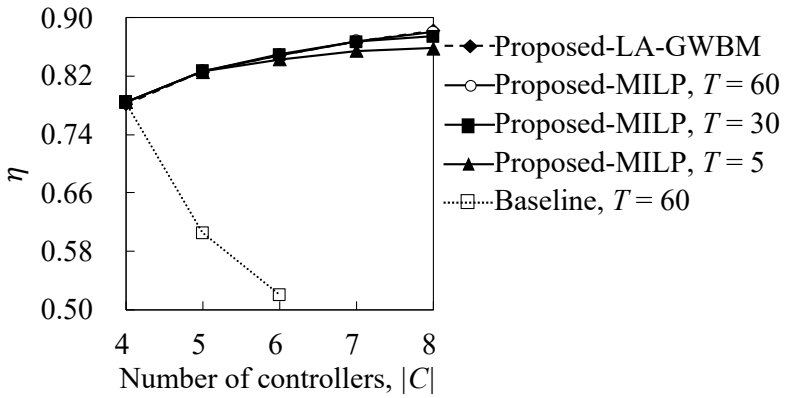


Figure 7.7: Comparison among values of η obtained by different models and approaches with limited admissible computation time.

7.5.3 Performance dependency

This work focuses on the MinAEL problem with considering a network with 50 switches and 10 controllers, or $|S| = 50$ and $|C| = 10$, to investigate the performance dependencies on other parameters. This work runs the heuristic of LA-GWBM to solve such a large problem. Unless specifically stated, parameters are set to the same as those in Section 7.5.2.

Figure 7.8 shows the dependencies on the controller capacity with different values of switch acceptable unavailabilities and controller failure probabilities. As the controller capacity increases, each switch can have a higher probability to connect controllers with low latency, which reduces the expected propagation latency for each switch. This work observes that the average-case expected propagation latency decreases as the acceptable unavailability of each switch increases. This is because less controllers are assigned to each switch when it has a less strict requirement for availability. On the contrary, for the same availability requirement, more controllers are required to be assigned for a switch when the failure probability of each controller increases. As a result, the average-case expected propagation latency increases.

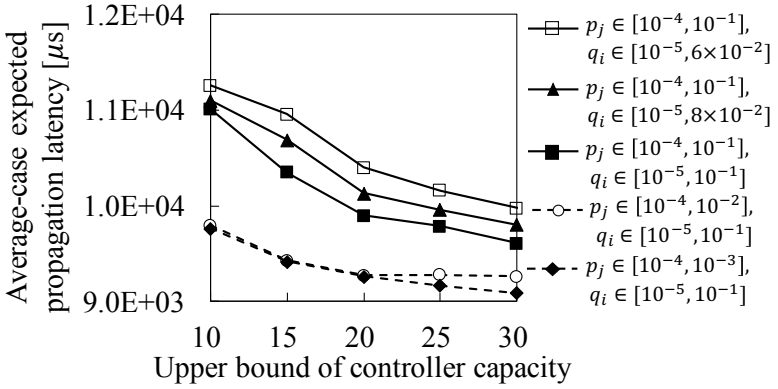


Figure 7.8: Dependencies on controller capacity with different values of controller failure probabilities and switch acceptable unavailabilities.

7.6 Chapter summary

This chapter proposed a master and slave controller assignment model against multiple controller failures with considering propagation latency between switches and controllers. A set of controllers is assigned to each switch to guarantee its survivability in a certain degree. This work considered three controller assignment problems optimizing the average-case expected propagation latency, the worst-case expected propagation latency, and the expected number of switches within a propagation latency bound, respectively. Given assigned controllers, the master controller in each failure case is determined based on LLFP, which is proven as the optimal policy in this paper. This work formulated three MILP problems for the proposed model with different goals. This work proved that all the three problems are NP-complete. This work developed a heuristic of LA-GWBM; this work showed that it provides a 1/2-approximation for the case without the survivability guarantee constraint. The numerical results revealed that the computation time of proposed model to obtain the optimal solution is about 10^2 times shorter than that of the baseline. For a small size problem, LA-GWBM obtains a solution, where the difference between the obtained objective value and the optimal value is less than 14% of the optimal value; its computation time is about 10^3 times shorter than that of solving the MILP problem. For a large size problem, LA-GWBM provides a better objective value than the MILP problem even setting the admissible computation time to solve the MILP problem as 10^3 times longer than that required by LA-GWBM.

Algorithm 12: Lower-bound aware greedy weighted bipartite b-matching

Input: q_i, $\forall i \in S$, $c_j, p_j, \forall j \in C$, and E
Ouput: C_i and E'
Set $U = S$, $T = C$, and $M = E$
Set $E' = \emptyset$
Set $C_i = \emptyset$ for each switch $i \in U$
while $U \neq \emptyset$, $T \neq \emptyset$, and $M \neq \emptyset$ **do**
  Assign controller $j \in T$ to switch $i \in U$, where $(i, j) \in M$ has optimal value for marginal gain, or set $C_i \leftarrow C_i \cup \{j\}$ and set $E' \leftarrow E' \cup \{(i, j)\}$
  Set $c_j = c_j - 1$
  Set $M \leftarrow M \setminus \{(i, j)\}$
  if $\prod_{j \in C_i} p_j \leq q_i$ **then**
    Set $U \leftarrow U \setminus \{i\}$
  end
  if $c_j = 0$ **then**
    Set $T \leftarrow T \setminus \{j\}$
    Set $M \leftarrow M \setminus \{(i', j) | (i', j) \in M\}$
  end
end
For MinAEL and MinWEL: **return**
For MaxENS:
while $T \neq \emptyset$ and $M \neq \emptyset$ **do**
  Assign controller $j \in T$ to switch $i \in S$, where $(i, j) \in M$ has optimal value for marginal gain, or set $C_i \leftarrow C_i \cup \{j\}$ and set $E' \leftarrow E' \cup \{(i, j)\}$
  Set $c_j = c_j - 1$
  Set $M \leftarrow M \setminus \{(i, j)\}$
  if $c_j = 0$ **then**
    Set $T \leftarrow T \setminus \{j\}$
    Set $M \leftarrow M \setminus \{(i', j) | (i', j) \in M\}$
  end
end
return

Chapter 8

Conclusions

While network virtualization brings a more flexible and efficient network, it makes network management such as resource allocation more challenging. In addition, the reliability of an environment with network virtualization has become a major concern. This thesis studied five specific problems about reliable resource allocation for network virtualization.

Firstly, this thesis proposed a primary and backup resource allocation model that provides a probabilistic protection guarantee for VMs against multiple PM failures to minimize the required total capacity. A PM in the cloud provider is used to accept both primary and backup resources. Considering the probabilistic protection guarantee in a general-capacity cloud provider leads to a nonlinear programing problem for primary and backup resource allocation against multiple failures. By adopting robust optimization with extensive mathematical operations, this work formulated the primary and backup resource allocation problem as an MILP problem, where capacity fragmentation is suppressed. This work proved that the primary and backup resource allocation problem is NP-hard by showing that the partition problem is reducible to it. For the problem with a large size, this work introduced the SA heuristic. Numerical results observed that about one-third of the total capacity is saved in the examined cases by adopting the proposed model. In addition, this work evaluated different models in dynamic scenarios, where both situations of the requested VMs arriving and the existing VMs releasing are considered. The results revealed that the proposed model outperforms the conventional models

in terms of both blocking probability and resource utilization.

Secondly, this thesis proposed a backup computing and transmission resource allocation model for VNs with the probabilistic protection against multiple facility node failures. Both backup computing and transmission resource allocations are considered to minimize the required backup computing capacity. Considering that the required backup transmission capacity can affect the required backup computing capacity, this work analyzed backup transmission resource sharing with multiple facility node failures based on graph theory. A heuristic algorithm was introduced to solve the problem. The results revealed that the proposed model outperforms the baseline in terms of both feasibility and required backup computing capacity. this work discussed the application scenarios for the proposed model with different degrees of backup transmission resource sharing. With the analyses, a network operator can consider an appropriate degree of backup transmission resource sharing based on practical requirements.

Thirdly, this thesis proposed a backup resource allocation model for middleboxes with considering both failure probabilities of network functions and backup servers. This work took the importance of functions into account by defining a weighted unavailability for each function. This work aimed to find an assignment of backup servers to functions where the worst weighted unavailability is minimized. This work formulated the BRAMI problem as an MILP problem. This work proved that the BRAMI problem is NP-complete. Two heuristics based on the greedy approach and one based on the LPR approach were introduced to solve the same optimization problem. The computational time complexities of three heuristic algorithms were analyzed as polynomials. This work analyzed the approximation performances of different heuristic algorithms by providing several lower and upper bounds. This work compared among the results obtained by different approaches and the lower and upper bounds. The results showed the pros and cons of different approaches. When the BRAMI problem becomes large, solving the MILP problem needs a long computation time to obtain the optimal solution, but a relatively much shorter time to obtain a solution comparable to the optimal one. Heuristic approaches outperform the MILP approach when the admissible computation time is set short. The CGA algorithm and the SGA algorithm provide less deviations and

require shorter computation time than the LPR-TBR algorithm does. However, only the performance of LPR-TBR algorithm has an upper bound and there is no approximation guarantee for the other two heuristics. Referring to the analyses, a network operator can choose an appropriate approach according to the requirements in specific application scenarios.

Fourthly, this thesis proposed an unavailability-aware backup allocation model with the shared protection for middleboxes with comprehensively considering heterogeneous procedures of functions and backup servers. The backup resources on a backup server can be shared by multiple functions. The proposed model aims to find the backup allocation for all functions to minimize the maximum unavailability among functions. This work developed an analytical approach based on the queueing theory to estimate the unavailability of middleboxes for a given backup allocation. The heterogeneous failure, repair, recovery, and waiting procedures of functions and backup servers, which lead to several different states for each function and for the whole system, are considered in the queueing approach. This work analyzed what all system states are, how they transit from/to each other, and what the equilibrium-state probability of each system state is. Based on the analytical approach, the heuristic was introduced to solve the backup allocation problem. The performance dependencies on the backup server capacity, failure rate, average repair time, and average recovery time were evaluated. This work compared the proposed model with the baseline model. The results observed that, compared to the baseline model, the proposed unavailability-aware model reduces the maximum unavailability 16% in average in the examined scenarios.

Fifthly, this thesis proposed a master and slave controller assignment model against multiple controller failures with considering propagation latency between switches and controllers. A set of controllers is assigned to each switch to guarantee its survivability in a certain degree. This work considered three controller assignment problems optimizing the average-case expected propagation latency, the worst-case expected propagation latency, and the expected number of switches within a propagation latency bound, respectively. Given assigned controllers, the master controller in each failure case is determined based on LLFP, which is proven as the optimal policy in this thesis. This work formulated three MILP problems for the proposed model with different

goals. This work proved that all the three problems are NP-complete. This work developed a heuristic of LA-GWBM; this work showed that it provides a 1/2-approximation for the case without the survivability guarantee constraint. The numerical results revealed that the computation time of proposed model to obtain the optimal solution is about 10^2 times shorter than that of the baseline. For a small size problem, LA-GWBM obtains a solution, where the difference between the obtained objective value and the optimal value is less than 14% of the optimal value; its computation time is about 10^3 times shorter than that of solving the MILP problem. For a large size problem, LA-GWBM provides a better objective value than the MILP problem even setting the admissible computation time to solve the MILP problem as 10^3 times longer than that required by LA-GWBM.

The five proposed models studied five typical application scenarios of network virtualization with considering the corresponding properties, respectively. This work provided different approaches with theoretical analyses in each model. A network operator or service provider can select appropriate models with suitable approaches according to the specific requirements to achieve a flexible, cost-effective, and dependable network virtualization environment.

For future works, there can be two directions to extend the proposed models. First, this thesis considers the resource allocation with deterministic demanding capacity. The network traffic may fluctuate over time. The demanding capacity of a network element, such as a VM, network function, or switch, can vary within a certain range. We can extend the proposed models with incorporating the traffic fluctuation. Second, this thesis studies the independent failure in different models, such as assuming that all backup servers fail independently and each backup server fails independently of any function. The other direction to extend the proposed models is to consider the correlated failure, which frequently occurs among network elements.

Bibliography

[1] J. Turner and D. Taylor, "Diversifying the Internet," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2005, pp. 755–760.

[2] N. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, 2010.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[4] Z. Cai, X. Li, and J. Gupta, "Heuristics for provisioning services to workflows in XaaS clouds," *IEEE Trans. Services Comput.*, vol. 9, no. 2, pp. 250–263, 2016.

[5] E. Oki, R. Kaneko, N. Kitsuwan, T. Kurimoto, and S. Urushidani, "Cloud provider selection models for cloud storage services to satisfy availability requirements," in *Proc. ICNC*, 2017, pp. 244–248.

[6] S. Maguluri, R. Srikant, and L. Ying, "Heavy traffic optimal resource allocation algorithms for cloud computing clusters," *Performance Evaluation*, vol. 81, pp. 20–39, 2014.

[7] M. Hadji and D. Zeghlache, "Minimum cost maximum flow algorithm for dynamic resource allocation in clouds," in *Proc. IEEE CLOUD*, 2012, pp. 876–882.

[8] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: state-of-the-art and re-

search challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 2016.

[9] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE INFOCOM*, 2006, pp. 1–12.

[10] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, 2012.

[11] A. Tootoonchian and Y. Ganjali, "Hyperflow: a distributed control plane for OpenFlow," in *Proc. INM/WREN*, 2010, pp. 3–3.

[12] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for large-scale production networks," in *OSDI*, vol. 10, 2010, pp. 1–6.

[13] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 1, pp. 333–354, 2018.

[14] P. Institute. 2016 cost of data center outages. [Online]. Available: https://www.vertivco.com/globalassets/documents/reports/2016-cost-of-data-center-outages-11-11_51190_1.pdf

[15] I. Addo, S. Ahamed, and W. Chu, "A reference architecture for high-availability automatic failover between paas cloud providers," in *Proc. TSA*, 2014, pp. 14–21.

[16] J. Sherry, P. X. Gao, S. Basu, A. Panda, A. Krishnamurthy, C. Maciocco, M. Manesh, J. Martins, S. Ratnasamy, L. Rizzo, and S. Shenker, "Rollback-recovery for middleboxes," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 227–240, 2015.

[17] A. da Silva, P. Smith, A. Mauthe, and A. Schaeffer-Filho, "Resilience support in software-defined networking: a survey," *Comput. Netw.*, vol. 92, pp. 189–207, 2015.

[18] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, "Remus: High availability via asynchronous virtual machine replication," in *Proc. NSDI*, 2008, pp. 161–174.

[19] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Optimizing virtual backup allocation for middleboxes," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2759–2772, 2017.

[20] T. Sato, F. He, E. Oki, T. Kurimoto, and S. Urushidani, "Implementation and testing of failure recovery based on backup resource sharing model for distributed cloud computing system," in *Proc. IEEE 7th Int. Conf. Cloud Netw. (CloudNet)*, 2018, pp. 1–3.

[21] W. Grover, *Mesh-based survivable transport networks: options and strategies for optical, MPLS, SONET and ATM networking*. Prentice Hall PTR, 2003.

[22] R. Potharaju and N. Jain, "Demystifying the dark side of the middle: a field study of middlebox failures in datacenters," in *Proc. ACM IMC*, 2013, pp. 9–22.

[23] B. Yang, Z. Xu, W. Chai, W. Liang, D. Tuncer, A. Galis, and G. Pavlou, "Algorithms for fault-tolerant placement of stateful virtualized network functions," in *Proc. IEEE ICC*, May 2018, pp. 1–7.

[24] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Designing optimal middlebox recovery schemes with performance guarantees," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2373–2383, 2018.

[25] H. Moens and F. D. Turck, "Customizable function chains: Managing service chain variability in hybrid NFV networks," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 711–724, 2016.

[26] J. Fan, M. Jiang, O. Rottenstreich, Y. Zhao, T. Guan, R. Ramesh, S. Das, and C. Qiao, "A framework for provisioning availability of NFV in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2246–2259, 2018.

[27] O. N. Foundation. Openflow switch specification version 1.5.1 (protocol version 0x06). [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf

[28] T. Das, V. Sridharan, and M. Gurusamy, "A survey on controller placement in SDN," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 1, pp. 472–503, 2019.

[29] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. HotSDN*, 2012, pp. 7–12.

[30] T. Yuan, X. Huang, M. Ma, and J. Yuan, "Balance-based SDN controller placement and assignment with minimum weight matching," in *Proc. IEEE ICC*, 2018, pp. 1–6.

[31] D. Suh and S. Pack, "Low-complexity master controller assignment in distributed SDN controller environments," *IEEE Commun. Lett.*, vol. 22, no. 3, pp. 490–493, 2018.

[32] T. Hu, Z. Guo, J. Zhang, and J. Lan, "Adaptive slave controller assignment for fault-tolerant control plane in software-defined networking," in *Proc. IEEE ICC*, 2018, pp. 1–6.

[33] M. Tanha, D. Sajjadi, and J. Pan, "Enduring node failures through resilient controller placement for software defined networks," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2016, pp. 1–7.

[34] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 3, pp. 991–1005, 2018.

[35] N. Perrot and T. Reynaud, "Optimal placement of controllers in a resilient SDN architecture," in *Proc. DRCN*, 2016, pp. 145–151.

[36] R. Banner and A. Orda, "Designing low-capacity backup networks for fast restoration," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[37] F. He, T. Sato, B. C. Chatterjee, T. Kurimoto, S. Urushidani, and E. Oki, "Robust optimization model for backup resource allocation in cloud provider," in *Proc. IEEE ICC*, 2018, pp. 1–6.

[38] F. He, T. Sato, and E. Oki, "Backup resource allocation model for virtual networks with probabilistic protection against multiple facility node failures," in *Proc. DRCN*, Mar. 2019, pp. 1–6.

[39] M. Johnston, H. Lee, and E. Modiano, "A robust optimization approach to backup network design with random failures," in *Proc. IEEE INFOCOM*, 2011, pp. 1512–1520.

[40] ——, "A robust optimization approach to backup network design with random failures," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1216–1228, Aug. 2015.

[41] J. Chu and C. Lea, "Optimal link weights for ip-based networks supporting hose-model vpns," *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 778–788, 2009.

[42] M. Kodialam, T. Lakshman, J. Orlin, and S. Sengupta, "Preconfiguring ip-over-optical networks to handle router failures and unpredictable traffic," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 5, pp. 934–948, 2007.

[43] I. Ouedraogo and E. Oki, "A green and robust optimization strategy for energy saving against traffic uncertainty," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1405–1416, 2016.

[44] Y. Zhu, Y. Liang, Q. Zhang, X. Wang, P. Palacharla, and M. Sekiya, "Reliable resource allocation for optically interconnected distributed clouds," in *Proc. IEEE ICC*, 2014, pp. 3301–3306.

[45] ——, "Reliable resource allocation with weighted srgs for optically interconnected clouds," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2014, pp. 2186–2191.

[46] S. Rajagopalan, D. Williams, and H. Jamjoom, "Pico replication: A high availability framework for middleboxes," in *Proc. 4th Annu. Symp. Cloud Comput.*, Oct. 2013.

[47] Y. Harchol, D. Hay, and T. Orenstein, "Ftvnf: Fault tolerant virtual network functions," in *Proc. 2018 Symp. Archit. Netw. Commun. Syst.*, Jul. 2018.

[48] F. He, T. Sato, and E. Oki, "Optimization model for backup resource allocation in middleboxes," in *Proc. IEEE 7th Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2018, pp. 1–6.

[49] ——, "Optimization model for backup resource allocation in middleboxes with importance," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1742–1755, Aug. 2019.

[50] F. He and E. Oki, "Unavailability-aware shared virtual backup allocation model for middleboxes," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, Apr. 2020.

[51] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[52] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *Proc. IEEE IWQoS*, Jun. 2017, pp. 1–10.

[53] L. Qu, C. Assi, K. Shaban, and M. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in NFV-enabled enterprise datacenter networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 554–568, Jul. 2017.

[54] L. Qu, M. Khabbaz, and C. Assi, "Reliability-aware service chaining in carrier-grade softwarized networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 558–573, Mar. 2018.

[55] L. Qu, C. Assi, M. Khabbaz, and Y. Ye, "Reliability-aware service function chaining with function decomposition and multipath routing," *IEEE Trans. Netw. Service Manag.*, 2019.

[56] D. Li *et al.*, "Availability aware VNF deployment in datacenter through shared redundancy and multi-tenancy," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1651–1664, 2019.

[57] J. Kleinberg and E. Tardos, *Algorithm design*. Boston: Pearson Education Inc., 2006.

[58] R. Graham, "Bounds for certain multiprocessing anomalies," *Bell Syst. Tech. J.*, vol. 45, no. 9, pp. 1563–1581, Nov. 1966.

[59] ——, "Bounds on multiprocessing timing anomalies," *SIAM J. Appl. Math.*, vol. 17, no. 2, pp. 416–429, Mar. 1969.

[60] J. Lenstra, D. Shmoys, and E. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Math. Program., Ser. A*, vol. 46, no. 1-3, pp. 259–271, Jan. 1990.

[61] D. Hochbaum and D. Shmoys, "Using dual approximation algorithms for scheduling problems theoretical and practical results," *J. ACM*, vol. 34, no. 1, pp. 144–162, Jan. 1987.

[62] N. Bansal and M. Sviridenko, "The santa claus problem," in *Proc. 38th Annu. ACM Symp. Theory Comput.*, May 2006, pp. 31–40.

[63] I. Bezakova and V. Dani, "Allocating indivisible goods," *ACM SIGecom Exchanges*, vol. 5, no. 3, pp. 11–18, Apr. 2015.

[64] B. Chatterjee, F. He, E. Oki, A. Fumagalli, and N. Yamanaka, "A span power management scheme for rapid lightpath provisioning and releasing in multi-core fiber networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 734–747, Apr. 2019.

[65] Y. Lai, A. Ali, M. Hossain, and Y. Lin, "Performance modeling and analysis of TCP and UDP flows over software defined networks," *J. Netw. Comput. Appl.*, vol. 130, pp. 76–88, Mar. 2019.

[66] R. Gouareb *et al.*, "Virtual network functions routing and placement for edge cloud latency minimization," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2346–2357, 2018.

[67] S. Agarwal *et al.*, "VNF placement and resource allocation for the support of vertical services in 5G networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 433–446, 2019.

[68] F. Malandrino *et al.*, "Reducing service deployment cost through VNF sharing," *IEEE/ACM Trans. Netw.*, vol. 27, no. 6, pp. 2363–2376, 2019.

[69] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "Resilience of SDNs based on active and passive replication mechanisms," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Dec. 2013, pp. 2188–2193.

[70] E. Spalla, D. Mafioletti, A. Liberato, G. Ewald, C. Rothenberg, L. Camargos, R. Villaca, and M. Martinello, "Ar2c2: Actively replicated controllers for SDN resilient control plane," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, Apr. 2016, pp. 189–196.

[71] F. He, T. Sato, and E. Oki, "Master and slave controller assignment model against multiple failures in software defined network," in *Proc. IEEE ICC*, May 2019, pp. 1–6.

[72] F. Ros and P. Ruiz, "Five nines of southbound reliability in software-defined networks," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, Aug. 2014, pp. 31–36.

[73] ——, "On reliable controller placements in software-defined networks," *Comput. Commun.*, vol. 77, pp. 41–51, Mar. 2016.

[74] B. Killi and S. Rao, "Optimal model for failure foresight capacitated controller placement in software-defined networks," *IEEE Commun. Lett.*, vol. 20, no. 6, pp. 1108–1111, Apr. 2016.

[75] B. Killi *et al.*, "Capacitated next controller placement in software defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 514–527, 2017.

[76] B. Killi and S. Rao, "Towards improving resilience of controller placement with minimum backup capacity in software defined networks," *Comput. Netw.*, vol. 149, pp. 102–114, Feb. 2019.

[77] C. Pham, D. Nguyen, N. Tran, K. Nguyen, and M. Cheriet, "Dynamic controller/switch mapping in virtual networks service chains," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[78] T. Wang *et al.*, "An efficient online algorithm for dynamic SDN controller assignment in data center networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2788–2801, 2017.

[79] S. Bera, S. Misra, and N. Saha, "Traffic-aware dynamic controller assignment in SDN," *IEEE Trans. Commun.*, 2020.

[80] L. Zhang, Y. Wang, X. Zhong, W. Li, and S. Guo, "Resource-saving replication for controllers in multi controller SDN against network failures," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, Apr. 2018, pp. 1–7.

[81] V. Sridharan, M. Gurusamy, and T. Truong-Huu, "On multiple controller mapping in software defined networks with resilience constraints," *IEEE Commun. Lett.*, vol. 21, no. 8, pp. 1763–1766, Apr. 2017.

[82] T. Hu, P. Yi, Z. Guo, J. Lan, and Y. Hu, "Dynamic slave controller assignment for enhancing control plane robustness in software-defined networks," *Future Gener. Comp. Sy.*, vol. 95, pp. 681–693, Jun. 2019.

[83] Z. Guo, W. Feng, S. Liu, W. Jiang, Y. Xu, and Z. Zhang, "Retroflow: maintaining control resiliency and flow programmability for software-defined wans," in *Proc. IEEE/ACM Int. Symp. Qual. Service (IWQoS)*, Jun. 2019, pp. 1–10.

[84] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions-i," *Math. Prog.*, vol. 14, no. 1, pp. 265–294, Dec. 1978.

[85] M. Fisher, G. Nemhauser, and L. Wolsey, "An analysis of approximations for maximizing submodular set functions-ii," *Polyhedral Combinatorics*, pp. 73–87, 1978.

[86] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrak, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1740–1766, 2011.

[87] Gurobi. Gurobi optimizer 9.0. [Online]. Available: http://www.gurobi.com

[88] IBM. Ibm ilog cplex optimization studio. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio

[89] D. Bertsimas and M. Sim, "The price of robustness," *Operations Research*, vol. 52, no. 1, pp. 35–53, 2004.

[90] R. Karp, *Complexity of Computer Computations*. New York: Miller, R. E. and J. W. Thatcher Eds. Plenum Press, 1972, ch. Reducibility among combinatorial problems, pp. 85–104.

[91] T. Segaram, *Programming collective intelligence: building smart web 2.0 applications*. O'Reilly Media, Inc, 2007.

[92] H. Khazaei, J. Misic, and V. Misic, "A fine-grained performance model of cloud computing centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2138–2147, 2013.

[93] D. Serrano, S. Bouchenak, Y. Kouki, T. Ledoux, J. Lejeune, J. Sopena, L. Arantes, and P. Sens, "Towards QoS-oriented SLA guarantees for online cloud services," in *Proc. IEEE/ACM CCGrid*, 2013, pp. 50–57.

[94] P. Brebner, "Is your cloud elastic enough?: performance modelling the elasticity of infrastructure as a service (iaas) cloud applications," in *Proc. ACM/SPEC ICPE*, 2012, pp. 263–266.

[95] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comp. Sy.*, vol. 28, no. 5, pp. 755–768, 2012.

[96] Q. Zhang, Q. Zhu, and R. Boutaba, "Dynamic resource allocation for spot markets in cloud computing environments," in *Proc. IEEE UCC*, 2011, pp. 178–185.

[97] K. Xiong and H. Perros, "Service performance and analysis in cloud computing," in *Proc. Services I*, 2009, pp. 693–700.

[98] N. Shahriar, S. Chowdhury, R. Ahmed, A. Khan, S. Fathi, R. Boutaba, and L. Liu, "Virtual network survivability through joint spare capacity allocation and embedding," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 502–518, Mar. 2018.

[99] S. Ayoubi, Y. Chen, and C. Assi, "Towards promoting backup-sharing in survivable virtual network design," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 3218–3231, Oct. 2016.

[100] H. Yu, V. Anand, C. Qiao, and G. Sun, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," in *Proc. IEEE ICC*, Jun. 2011, pp. 1–6.

[101] R. Luce and A. Perry, "A method of matrix analysis of group structure," *Psychometrika*, vol. 14, no. 2, pp. 95–116, Jun. 1949.

[102] J. Moon and L. Moser, "On cliques in graphs," *Israel J. Math.*, vol. 3, no. 1, pp. 23–28, Mar. 1965.

[103] J. Orlin, "A polynomial time primal network simplex algorithm for minimum cost flows," *Math. Prog.*, vol. 78, no. 2, pp. 109–129, Aug. 1997.

[104] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Commun. ACM*, vol. 78, no. 2, pp. 109–129, Aug. 1997.

[105] M. Batayneh, D. A. Schupke, M. Hoffmann, A. Kirstaedter, and B. Mukherjee, "On routing and transmission-range determination of multi-bit-rate signals over mixed-line-rate WDM optical networks for carrier ethernet," *IEEE/ACM Trans. Netw.*, vol. 19, no. 5, pp. 1304–1316, Oct. 2011.

[106] T-NOVA. Network functions implementation and testing - interim. [Online]. Available: http://www.t-nova.eu/wp-content/uploads/2016/03/TNOVA_D5.31-Network-Functions-Implementation-and-Testing-Interim_v1.0.pdf

[107] E. I. S. G. (ISG). Network functions virtualisation (NFV): Architectural framework. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf

[108] J. Herrera and J. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518–532, Aug. 2016.

[109] T. Korikawa, A. Kawabata, F. He, and E. Oki, "Carrier-scale packet processing system using interleaved 3d-stacked dram," in *Proc. IEEE ICC*, May 2018, pp. 1–6.

[110] S. Gerke, K. Panagiotou, J. Schwartz, , and A. Steger, "Maximizing the minimum load for random processing times," *ACM Trans. Algorithms*, vol. 11, no. 3, pp. 1–19, Jan. 2015.

[111] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 10, pp. 146–160, Jun. 1972.

[112] J. Hopcroft and R. Tarjan, "Algorithm 447: efficient algorithms for graph manipulation," *Commun. ACM*, vol. 16, no. 6, pp. 372–378, Jun. 1973.

[113] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: network processing as a cloud service," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 13–24, Aug. 2012.

[114] T. Kuwabara, Y, Mitsunaga, and H. Koga, "Calculation method of failure probabilities of optical fiber," *J. Lightw. Technol.*, vol. 11, no. 7, pp. 1132–1138, Jul. 1993.

[115] S. Sekigawa, S. Okamoto, N. Yamanaka, and E. Oki, "Expected capacity guaranteed routing based on dynamic link failure prediction," in *Proc. ICNC*, Feb. 2019, pp. 170–174.

[116] J. Little, "A proof for the queuing formula: $l = \lambda w$," *Oper. Res.*, vol. 9, no. 3, pp. 383–387, 1961.

[117] A. Bjorck, *Numerical methods in matrix computations.* Cham, Switzerland: Springer, 2015.

[118] F. Ahmed, J. Dickerson, and M. Fuge, "Diverse weighted bipartite b-matching," in *Proc. IJCAI*, 2017, pp. 35–41.

[119] C. Chen, L. Zheng, V. Srinivasan, A. Thomo, K. Wu, and A. Sukow, "Conflict-aware weighted bipartite b-matching and its application to e-commerce," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1475–1488, Jun. 2016.

[120] C. Chen, S. Chester, V. Srinivasan, K. Wu, and A. Thomo, "Group-aware weighted bipartite b-matching," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, pp. 459–468.

[121] J. Dickerson, K. Sankararaman, A. Srinivasan, and P. Xu, "Balancing relevance and diversity in online bipartite matching via submodularity," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 1877–1884.

[122] H. Gabow, "An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems," in *Proc. 15th Annu. ACM Symp. Theory Comput.*, Dec. 1983, pp. 448–456.

[123] K. Fujii, "Faster approximation algorithms for maximizing a monotone submodular function subject to a b-matching constraint," *Inf. Process. Lett.*, vol. 116, no. 9, pp. 578–584, Sep. 2016.

Publication List

Journal Papers

1. **F. He**, T. Sato, and E. Oki, "Optimization model for backup resource allocation in middleboxes with importance," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1742-1755, 2019.

2. R. Kang, **F. He**, T. Sato, and E. Oki, "Virtual network function allocation to maximize continuous available time of service function chains with availability schedule," *IEEE Transactions on Network and Service Management*, 2020. [Accepted]

3. T. Korikawa, A. Kawabata, **F. He**, and E. Oki, "Packet processing architecture with off-chip last level cache using interleaved 3D-stacked DRAM," *IEICE Transactions on Communications*, 2020. [Accepted]

4. T. Sato, **F. He**, E. Oki, T. Kurimoto, and S. Urushidani, "Experiment and availability analytical model of cloud computing system based on backup resource sharing and probabilistic protection guarantee," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 700-712, 2020.

5. T. Sawa, **F. He**, A. Kawabata, and E. Oki, "Algorithms for distributed server allocation problem," *IEICE Transactions on Communications*, 2020. [Accepted]

6. T. Korikawa, A. Kawabata, **F. He**, and E. Oki, "Packet processing architecture using last-level-cache slices and interleaved 3D-stacked DRAM," *IEEE Access*, vol. 8, pp. 59290-59304, 2020.

7. R. Fujita, **F. He**, T. Sato, and E. Oki, "Shared backup resource assignment for middleboxes," *Optical Switching and Networking*, vol. 37, 2020.

8. Y. Hirano, **F. He**, T. Sato, and E. Oki, "Backup network design against multiple link failures to avoid link capacity overestimation," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1254-1267, 2020.

9. Y. Zhang, **F. He**, T. Sato, and E. Oki, "Network service scheduling with resource sharing and preemption," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 764-778, 2020.

10. T. Sawa, **F. He**, T. Sato, B.C. Chatterjee, and E. Oki, "Defragmentation with reroutable backup paths in toggled 1+1 protection elastic optical networks," *IEICE Transactions on Communications*, vol. E103.B , no. 3, pp. 211-223, 2020.

11. T. Korikawa, A. Kawabata, **F. He**, and E. Oki, "Carrier-scale packet processing system using interleaved 3D-stacked DRAM," *IEEE Access*, vol. 7, pp. 75500-75514, 2019.

12. B.C. Chatterjee, **F. He**, E. Oki, A. Fumagalli, and N. Yamanaka, "A span power management scheme for rapid lightpath provisioning and releasing in multi-core fiber networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 734-747, 2019.

International Conference Papers

1. **F. He** and E. Oki, "Load balancing model against multiple controller failures in software defined networks," in *Proceedings of IEEE International Conference on Communications (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1-6.

2. **F. He** and E. Oki, "Unavailability-aware shared virtual backup allocation model for middleboxes," in *Proceedings of IEEE/IFIP Network*

Operations and Management Symposium (NOMS), Budapest, Hungary, Apr. 2020, pp. 1-7.

3. **F. He**, T. Sato, and E. Oki, "Survivable virtual network embedding model with shared protection over elastic optical network," in *Proceedings of IEEE 7th International Conference on Cloud Networking (CloudNet)*, Coimbra, Portugal, Nov. 2019, pp. 1-3.

4. **F. He**, T. Sato, and E. Oki, "Probabilistic protection model for virtual networks against multiple facility node failures," in *Proceedings of 15th International Conference on IP + Optical Network (iPOP)*, Kanagawa, Japan, May 2019.

5. **F. He**, T. Sato, and E. Oki, "Master and slave controller assignment model against multiple failures in software defined network," in *Proceedings of IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019, pp. 1-6.

6. **F. He**, T. Sato, and E. Oki, "Backup resource allocation model for virtual networks with probabilistic protection against multiple facility node failures," in *Proceedings of 15th International Conference on the Design of Reliable Communication Networks (DRCN)*, Coimbra, Portugal, Mar. 2019, pp. 37-42.

7. **F. He**, T. Sato, and E. Oki, "Optimization model for backup resource allocation in middleboxes," in *Proceedings of IEEE 7th International Conference on Cloud Networking (CloudNet)*, Tokyo, Japan, Oct. 2018, pp. 1-6.

8. **F. He**, T. Sato, B.C. Chatterjee, T. Kurimoto, S. Urushidani, and E. Oki, "Robust optimization model for backup resource allocation in cloud provider," in *Proceedings of IEEE International Conference on Communications (ICC)*, Kansas City, USA, May 2018, pp. 1-6.

9. M. Zhu, **F. He**, and E. Oki, "Multiple backup resource allocation with workload-dependent failure probability," in *Proceedings of IEEE Global Communications Conference (Globecom)*, Taipei, Taiwan, Dec. 2020.

10. E. Oki, T. Sawa, **F. He**, T. Sato, and B. C. Chatterjee, "Performance of hitless defragmentation with rerouting for quasi 1+1 protected elastic optical networks," in *International Conference on Transparent Optical Networks (ICTON 2020)*, Bari, Italy, Jul. 2020. (Invited paper)

11. R. Fujita, **F. He**, and E. Oki, "Shared backup resource assignment for middleboxes considering server capability," in *Proceedings of 20th IEEE International Conference on High Performance Switching and Routing (HPSR)*, Newark, USA, May 2020, pp. 1-6.

12. S. Masuda, **F. He**, A. Kawabata, and E. Oki, "Distributed Server Allocation Model with Preventive Start-Time Optimization Against Single Failure," in *Proceedings of 20th IEEE International Conference on High Performance Switching and Routing (HPSR)*, Newark, USA, May 2020, pp. 1-6.

13. Y. Zhang, **F. He** and E. Oki, "Network service mapping and scheduling under uncertain processing time," in *Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Budapest, Hungary, Apr. 2020.

14. R. Kang, **F. He** T. Sato, and E. Oki, "Demonstration of network service header based service function chain application with function allocation model," in *Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Budapest, Hungary, Apr. 2020.

15. M. Ito, **F. He**, and E. Oki, "Robust optimization model for probabilistic protection under uncertain virtual machine capacity in cloud," in *Proceedings of 16th International Conference on the Design of Reliable Communication Networks (DRCN)*, Milan, Italy, Mar. 2020, pp. 1-8.

16. Y. Hirano, **F. He**, T. Sato, and E. Oki, "Preventive start-time optimization to determine link weights against multiple link failures," in *Proceedings of IEEE 7th International Conference on Cloud Networking (CloudNet)*, Coimbra, Portugal, Nov. 2019, pp. 1-3.

17. T. Sawa, **F. He**, A. Kawabata, and E. Oki, "Polynomial-time algorithm for distributed server allocation problem," in *Proceedings of IEEE 7th International Conference on Cloud Networking (CloudNet)*, Coimbra, Portugal, Nov. 2019, pp. 1-3.

18. R. Kang, **F. He**, T. Sato, and E. Oki, "Virtual network function allocation to maximize continuous available time of service function chains," in *Proceedings of IEEE 7th International Conference on Cloud Networking (CloudNet)*, Coimbra, Portugal, Nov. 2019, pp. 1-6.

19. T. Sawa, **F. He**, T. Sato, B.C. Chatterjee, and E. Oki, "Defragmentation considering link congestion in toggled 1+1 path protected elastic optical networks," in *Proceedings of 24th OptoElectronics and Communications Conference/Photonics in Switching and Computing (OECC/PSC)*, Fukuoka, Japan, Jul. 2019, pp. 1-3.

20. Y. Zhang, **F. He**, T. Sato, and E. Oki, "Optimization of network service scheduling with resource sharing and preemption," in *Proceedings of 19th IEEE International Conference on High Performance Switching and Routing (HPSR)*, Xi'an, China, May 2019, pp. 1-6.

21. T. Korikawa, A. Kawabata, **F. He**, and E. Oki, "Packet processing architecture with off-chip llc using interleaved 3D-stacked DRAM," in *Proceedings of 19th IEEE International Conference on High Performance Switching and Routing (HPSR)*, Xi'an, China, May 2019, pp. 1-6.

22. R. Fujita, **F. He**, T. Sato, and E. Oki, "Optimization of backup resource assignment for middleboxes," in *Proceedings of 19th IEEE International Conference on High Performance Switching and Routing (HPSR)*, Xi'an, China, May 2019, pp. 1-6.

23. Y. Zhang, **F. He**, T. Sato, and E. Oki, "Flexible scheduling approach for network services in virtual networks," in *Proceedings of 15th International Conference on IP + Optical Network (iPOP)*, Kanagawa, Japan, May 2019.

24. T. Sawa, **F. He**, T. Sato, B.C. Chatterjee, and E. Oki, "Defragmentation using reroutable backup paths in toggled 1+1 path protected elastic optical networks," in *Proceedings of 24th Asia-Pacific Conference on Communications (APCC)*, Ningbo, China, Nov. 2018, pp. 422-427.

25. T. Sato, **F. He**, E. Oki, T. Kurimoto, and S. Urushidani, "Implementation and testing of failure recovery based on backup resource sharing model for distributed cloud computing system," in *Proceedings of IEEE 7th International Conference on Cloud Networking (CloudNet)*, Tokyo, Japan, Oct. 2018, pp. 1-3.

26. Y. Hirano, **F. He**, T. Sato, and E. Oki, "Backup network design scheme for multiple link failures to avoid overestimating link capacity," in *Proceedings of 18th IEEE International Conference on High Performance Switching and Routing (HPSR)*, Bucharest, Romania, Jun. 2018, pp. 1-6.

27. T. Korikawa, A. Kawabata, **F. He**, and E. Oki, "Carrier-scale packet processing system using interleaved 3D-stacked DRAM," in *Proceedings of IEEE International Conference on Communications (ICC)*, Kansas City, USA, May 2018, pp. 1-6.

Technical Reports and Local Conference Papers

1. **F. He**, T. Sato, and E. Oki, "Backup resource allocation model against multiple controller failures in software defined network," at *Photonic Network Workshop*, Otaru, Japan, Aug. 2019.

2. **F. He**, T. Sato, B.C. Chatterjee, T. Kurimoto, S. Urushidani, and E. Oki, "Mix integer linear programming model for backup resource allocation in cloud provider," at *Photonic Network Workshop*, Kobe, Japan, Jul. 2018.

Awards

1. IEEE ComSoc Student Grant at IEEE International Conference on Communications (ICC) in 2020
2. Excellent Paper Award at IEEE International Conference on High Performance Switching and Routing (HPSR) in 2019